Technische Universität Braunschweig
Carl-Friedrich-Gauß-Fakultät
Institut für Betriebssysteme und Rechnerverbund

Vom Körper zum Server: Mobile und drahtlose Datenerfassung und -übertragung in Gesundheitspflege-, Notfall- und AAL-Szenarien

Von der
Carl-Friedrich-Gauß-Fakultät
der Technischen Universität Carolo-Wilhelmina zu Braunschweig

zur Erlangung des Grades eines

Doktoringenieurs (Dr.-Ing.)

genehmigte Dissertation

von
Felix Büsching
geboren am 21.04.1977
in Braunschweig

Eingereicht am: 24.10.2013
Disputation am: 27.11.2013
1. Referent: Prof. Dr.-Ing. Lars Wolf
2. Referent: Prof. Dr. Stefan Fischer

2013

Kurzfassung

In vielen Szenarien in den Bereichen Gesundheitspflege, Notfallversorgung und Ambient Assisted Living (AAL) werden Daten durch Sensoren am menschlichen Körper erhoben. Diese Daten werden je nach Anwendungsfall entweder zur späteren Auswertung lokal gespeichert oder aber direkt über eine Funkverbindung zu einer Datensenke gesendet. In dieser Arbeit wird ein System vorgestellt, das in der Lage ist, Szenarien mit unterschiedlichen Anforderungen zu ermöglichen, wobei die gesamte Übertragungsstrecke vom Körper über häusliche Gateways bis hin zu Servern bzw. Backendsystemen abgedeckt wird. Dabei stellt sich zunächst die Herausforderung, inwiefern die Daten sicher, zuverlässig und verlustfrei vom aufnehmenden Sensorsystem am menschlichen Körper hin zu einer verarbeitenden Basisstation innerhalb der Wohnung der betreffenden Person übertragen werden können. Ferner gilt es auch, die dort lagernden Daten verlässlich zu sichern und die Betriebssicherheit dieser Basisstationen durch eine Fernüberwachung mittels zentraler Instanzen zu ermöglichen, welche dabei auch datenschutzrelevante Anforderungen erfüllen muss.

Nach einer Analyse der im Umfeld anzutreffenden Anwendungsfälle und der dort eingesetzten Sensoren wird das System zunächst in Bezug auf die aufnehmenden Sensoren und die daraus resultierenden Datenraten dimensioniert. Da kein geeignetes und die Bedürfnisse erfüllendes Hardwaresystem verfügbar war, wurde mit INGA ein Sensorknoten geschaffen, der den Anforderungen an die Sensorausstattung, die Verarbeitungskapazität und die Funkübertragung gerecht wird. Mithilfe dieses neu entwickelten Sensorknotens werden dabei Aktivitäts- und Bewegungsdaten mit verschiedenen Sensoren aufgezeichnet. Dieses Datenaufkommen wird anschließend mittels einer für den Anwendungsfall angepassten Deltakodierung effizient und verlustfrei reduziert. Zur Absicherung der Funkübertragung werden die Daten mit einem auf One-Time-Pads basierenden Kryptosystem verschlüsselt.

Die Übertragung zum häuslichen Gateway geschieht über ein unterbrechungstolerantes Kommunikationsprotokoll, welches auch eine implizite Synchronisation derjenigen Daten ermöglicht, die außerhalb der Kommunikationsreichweite des Gateways, also außerhalb der Wohnung, aufgenommen wurden. Um im außerhäuslichen Bereich auch Notfallmeldungen absetzen zu können, wird außerdem ein mobiler Notfallkanal über ein Smartphone realisiert.

Die aufgezeichneten Daten werden in den betrachteten Szenarien auf dem Gateway in der Wohnung der betreffenden Person gespeichert und dort ausgewertet; Alarmmeldungen werden bei einem (durch das Gateway erkannten) Notfall durch selbiges ausgelöst. Um die Integrität dieser verteilen Gateways auch bei einer größeren Anzahl von Installationen sicherstellen zu können, wird ein System zur Fernüberwachung der verteilten Gateways über selbstinitiierte VPN-Verbindungen zu einer zentralen Überwachungsinstanz geschaffen. Über diesen gesicherten Kanal werden dann auch die Sicherungen der lokalen Daten zu einem Backend durchgeführt.

Mit den vorgestellten Lösungen werden somit vielfältige Anwendungsfälle mit teils gegensätzlichen Anforderungen unterstützt, die noch dazu parallel auf ein und denselben Sensorsystemen und mit den gleichen Protokollen betrieben werden können.

Abstract

In the area of healthcare, emergency response and Ambient Assisted Living (AAL), data are gathered by different sensors attached to the human body. Depending on the specific use case, these data are either stored locally for an offline analysis or transmitted directly via a radio link to a sink to enable an online assessment. In this thesis, a system that supports scenarios with different requirements is presented whereas the whole communication chain from the human body via home gateways to distant backend systems is addressed. Transferring sensor data from the worn sensor systems to a home gateway in a secure, safe, and lossless way is one of the major challenges. Once the data arrived at this home gateway, it has to be stored and backed up safely and reliably. Additionally, to ensure the correct and dependable functionality of these gateways, a remote monitoring and a backup to distant locations have to be provided; whereas theses remote systems have to ensure the demands for privacy and data security.

After analyzing specific use cases and particular sensors which are commonly used in the addressed application areas, the system was dimensioned regarding sensors and data rates. As no existing sensor system fulfilled the requirements completely, a new wireless sensor node has been developed: INGA. On the one hand, this node is equipped with the required set of sensors for movement and activity detection. On the other hand, the computation and communications capabilities are suitable for the addressed use cases.

The recorded sensor data are reduced by an adapted delta coding scheme in an efficient and lossless way. The wireless channel is secured by a cryptosystem relying on One-Time-Pads.

Data are transferred to the gateway by a Disruption Tolerant Network protocol which also allows an implicit synchronization of data recorded outside the communication range of the gateway. To additionally enable a personal emergency response system outside of the habitation, a smartphone is utilized for mobile alarm messages.

In the addressed scenarios, data are stored and processed on the home gateway in the residence of the user. In case of a detected emergency these gateways may trigger alarm messages. To ensure the integrity and functionality of the distributed gateways, a remote instance monitors these systems. The communication between the distributed systems and the central monitoring instance is achieved by a VPN connection which is initiated by the distributed gateways. This secure communication channel is additionally used for backups to a remote location.

The presented solutions support multiple use cases with competing aims which are now able to run simultaneously on the same sensor systems and using the same protocols.

Danksagung

„For those about to rock, we salute you!“

AC/DC, 1981

Während der fünf Jahre, in denen der niedersächsische Forschungsverbund zur Gestaltung Altersgerechter Lebenswelten (GAL) gemeinsam daran arbeitete, Szenarien zu entwerfen, Prototypen zu entwickeln und Technologien zu evaluieren, haben viele verschiedene Menschen mit diversen Professionen an unterschiedlichen Institutionen gemeinsam kleinere und größere Dinge geschaffen. So wurden auch die Arbeiten, die die Grundlagen für diese Dissertation bilden, nicht in Isolation getätigt: Basis für einen großen Teil der Tätigkeiten war ohne Zweifel das GAL-Projekt und die in diesem Zusammenhang gelebte interdisziplinäre Zusammenarbeit.

Besonders erwähnen möchte ich in diesem Kontext die Kollegen des PLRI in Braunschweig und Hannover und des OFFIS in Oldenburg: In zahlreichen Diskussionen mit den Kollegen Matthias Gietzelt, Klaus-Hendrik Wolf, Michael Marschollek, Marco Eichelberg, Axel Helmer und Enno Steen entstanden viele Anwendungsfälle und wurden viele Herausforderungen, die in dieser Arbeit dann behandelt wurden, überhaupt erst identifiziert.

Auch meine direkten Kollegen und Studenten am IBR scheuten keine Debatte und es wurden viele Ideen und deren Umsetzungen gemeinsam diskutiert, implementiert und evaluiert. So startete die Konzeption des INGA-Sensorknotens (Kapitel 5) mit einer Projektarbeit von Ulf Kulau und wurde seitdem von mehreren Mitarbeitern und Studenten vorangetrieben. Die Implementierung von μDTN (Abschnitt 6.1.3) basiert auf Arbeiten von Georg von Zengen und wurde von Wolf-Bastian Pöttner stetig erweitert und optimiert. Neben den bereits erwähnten Kollegen halfen auch etliche Diskussionen bei Kaffee oder Bier (unter anderen) mit Dieter Brökelmann, Jens Brandt, Kai Homeier, Oliver Wellnitz, Sebastian Schildt und Tobias Baumgartner, zu neuen Erkenntnissen zu gelangen oder zumindest von Rückschlägen abzulenken.

Markus Weißkopf und Britta Eisenbarth vom Haus der Wissenschaft in Braunschweig haben mich motiviert, das zwischenzeitlich Erarbeite auch immer mal wieder beim Science Slam einer breiteren Öffentlichkeit vorzustellen, was sowohl für die Reflexion sehr hilfreich war aber auch eine willkommene Ablenkung darstellte.

Trotz des engen Zeitplans hat sich Stefan Fischer als zweiter Referent nicht über meine teils chaotische Arbeitsweise beschwert; Reinhold Haux übernahm nicht nur die Rolle des Projektsprechers sondern auch den Vorsitz der Promotionskommission, wobei ich ihn in beiden Rollen sehr zu schätzen lernte.

Ein besonderer Dank gilt meinem Mentor Lars Wolf, der es mir ermöglicht hat, in dieser Arbeit an spannenden Themen zu forschen, selbstbestimmt Schwerpunkte zu setzen und der mir immer mit Anregungen, Kritik und Unterstützung zur Seite stand.

Meine Eltern Gudrun und Fritz Büsching sowie meine Frau Jana haben mich zwar nicht fachlich, jedoch menschlich und praktisch unterstützt, indem sie für mich da waren, teils unverständliche Texte korrigiert haben und mir in stressigen Zeiten einfach den Rücken freigehalten haben.

Allen Beteiligten danke ich von ganzem Herzen für die spannende Zeit, die geleisteten Beiträge und die großartige Unterstützung!

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

„I am a tiny, insignificant, ignorant bit of carbon.
I have one life, and it is short and unimportant
But thanks to recent scientific advances
I get to live twice as long
As my great great great great uncleses and auntses.
Twice as long to live this life of mine
Twice as long to love this wife of mine
Twice as many years of friends and wine
Of sharing curries and getting shitty
At good-looking hippies
With fairies on their spines
And butterflies on their titties."

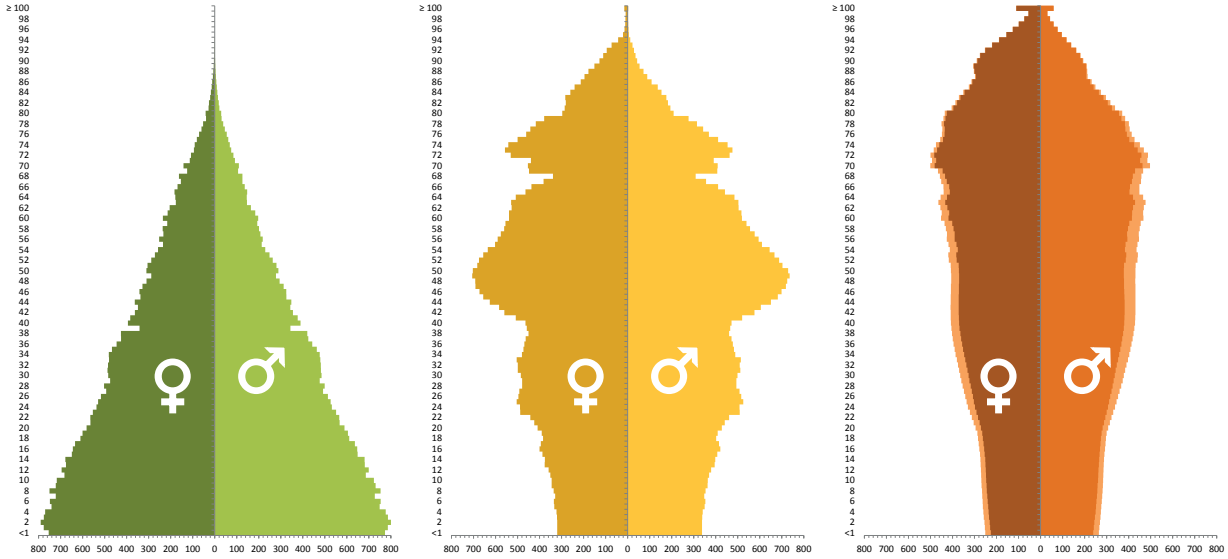
Tim Minchin – Storm, 2009[1]

Was Tim Minchin hier in einem Auszug aus seinem Gedicht *Storm* aus dem Jahr 2009 beschreibt ist allerdings nur die eine Seite der Medaille: Fortschreitende Entwicklungen in der Medizin, eine ausreichende Versorgung mit Nahrungsmitteln und Verbesserungen in Hygiene haben dazu geführt, dass die durchschnittliche Lebenserwartung in Deutschland und der sogenannten westlichen Welt heutzutage in etwa doppelt so hoch ist wie vor 120 Jahren. Da auf der anderen Seite aber auch ein signifikanter Rückgang der Geburtenrate zu verzeichnen ist, wird sich neben dem erreichbaren Alter auch die Altersstruktur der Gesellschaft nachhaltig verändern.

Dass die Gesellschaft älter wird und immer weniger Erwerbstätige immer mehr Rentnern und Pensionären gegenüberstehen, ist mittlerweile unbestritten. Zur Verdeutlichung ist dazu in Abbildung 1.1 die Entwicklung der Altersstruktur in Deutschland dargestellt: Im Jahr 1910 ist noch eine klassische Pyramidenform zu erkennen; der Anteil der jüngeren Menschen überwiegt den Anteil der älteren Menschen bei weitem und die Anzahl der Menschen höheren Alters nimmt in etwa linear ab. Während heutzutage schon ein klarer Trend zu einem insgesamt höheren Alter zu erkennen ist, wird aber gleichzeitig auch deutlich, dass sich die Proportionen der Struktur stark verändert haben. Auch heute schon ist die Anzahl der jüngeren Menschen verhältnismäßig klein im Vergleich zu der Anzahl der älteren Menschen. In der Zukunft stellt sich dann die einstige Pyramide quasi auf den Kopf und das durchschnittlich erreichbare Alter ist noch einmal deutlich erhöht, während gleichzeitig immer weniger Menschen geboren werden.

Zur weiteren Verdeutlichung ist in Abbildung 1.2 die Altersverteilung nach Altersgruppen aufgetragen. Hier lässt sich deutlich erkennen, dass gerade die Altersgruppe der über 65-jährigen am stärksten wächst. Während im Jahr 2013 der Anteil dieser Gruppe noch bei $\sim$ 21 % liegt, wächst er bis zum Jahr 2060 auf knapp 34 % an. In dieser Altersgruppe nimmt dann der Anteil der Personen, die 85 Jahre oder älter sind, am stärksten zu; dabei werden im Durchschnitt Frauen vier Jahre älter als Männer. In den stark individualisierten Gesellschaften der westlichen Welt führt diese Entwicklung dazu, dass auch immer mehr Menschen im Alter alleine leben. Nach Daten des Statistischen Bundesamts [1] liegt der Anteil der Alleinlebenden in Deutschland im Bundesdurchschnitt bei $\sim$ 19 % (EU-27-Durchschnitt $\sim$ 13 %); bei den 80-Jährigen liegt dieser Anteil in

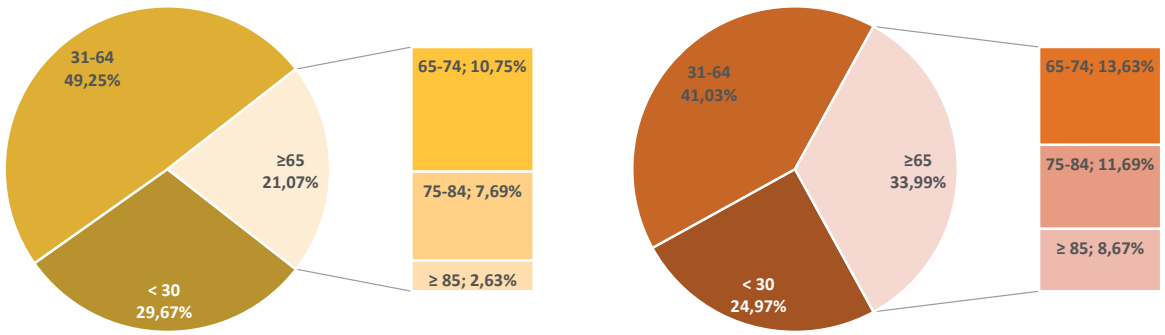
[1] http://youtu.be/HhGuXCuDb1U



(a) 1910 (b) 2013 – Prognose (c) 2060 – min/max Prognosen

Abbildung 1.1: Die Altersstruktur in Deutschland, getrennt nach Geschlechtern, aufgetragen in jeweils 1000 Menschen pro ganzzahliger Altersstufe.

Deutschland im Jahr 2011 bei den Männern bei 22 % und bei den Frauen bei 56 %. Jenseits der 85 Jahre leben knapp 75 % der Frauen alleine.



(a) 2013 (b) 2060

Abbildung 1.2: Die Altersverteilung nach Altersgruppen in Deutschland. Datenbasis jeweils: Statistisches Bundesamt[2].

Diese demographische Herausforderung, also die Entwicklung der Altersstruktur von dem, was früher normal war, zu dem, was morgen normal sein wird, ist wohl eine der zentralen Herausforderungen unserer Generation. Die Situation der alleinlebenden Älteren ist dabei ein zusätzliches Problemfeld, welches speziell in nordwesteuropäischen Ländern anzutreffen ist.

Da mit höherem Alter oft auch eine zunächst körperliche und später auch geistige Beeinträchtigung einhergeht, aber die Anzahl der Personen, die in Pflegeberufen arbeiten, nicht in gleichem Maße zunimmt, wie die der unterstützungsbedürftigen Menschen, wird in Zukunft immer häufiger versucht werden, diese Versorgungslücke mit technischen Hilfsmitteln zu schließen. Besonders wenn die technischen Hilfsmittel

[2] http://www.destatis.de

dabei unterstützen, die Eigenständigkeit des Individuums zu erhalten und es so beispielsweise einer Person erlauben, länger in gewohnter Umgebung zu leben, wird von einer erhöhten Nutzerakzeptanz ausgegangen. Für die Forschung in diesem Bereich hat sich der Terminus des *umgebungsunterstützten Lebens (Ambient Assisted Living (AAL))* etabliert.

Zahlreiche Forschungsprojekte setzen sich auf unterschiedlichen Ebenen mit dieser Herausforderung auseinander. Dabei werden teils gesellschaftliche Aspekte, teils technische Lösungen, teils wirtschaftliche Zusammenhänge oder medizinische Voraussetzungen fokussiert.

In dieser Arbeit soll es um technische Grundlagen gehen, die eine individuelle medizinische Versorgung von Menschen im Alter ermöglichen.

1.1 Gestaltung Altersgerechter Lebenswelten

Ein Großteil dieser Arbeit fand innerhalb des *Niedersächsischen Forschungsverbunds zur Gestaltung altersgerechter Lebenswelten – Informations- und Kommunikationstechnik zur Gewinnung und Aufrechterhaltung von Lebensqualität, Gesundheit und Selbstbestimmung in der zweiten Lebenshälfte (GAL)* [2] statt. Die erklärten Ziele des Forschungsverbunds sind es laut Vorhabensbeschreibung und Projektplan,

- neue Verfahren der Informations- und Kommunikationstechnik für altersgerechte Lebenswelten zu identifizieren, weiterzuentwickeln und zu evaluieren (inhaltliche Zielsetzung) sowie
- die einschlägigen niedersächsischen Forschungseinrichtungen in die Lage zu versetzen, sich federführend an größeren nationalen oder internationalen Forschungsvorhaben zu dieser Thematik beteiligen zu können (forschungsstrategische Zielsetzung).

An diesen Fragestellungen arbeiteten im Forschungsprojekt mehr als 80 Wissenschaftler von insgesamt 15 verschiedenen Institutionen, weshalb an dieser Stelle nur auf einige technische Teilaspekte eingegangen werden kann. Eine Übersicht aller im Projekt entstandenen Publikationen findet sich auf den Projektwebseiten[3]; der bisher noch nicht veröffentlichte Abschlussbericht ist in naher Zukunft ebenfalls dort abrufbar.

Der Fokus des Projekts liegt – entsprechend der gerade beschriebenen Ausgangslage in der Bundesrepublik Deutschland – auf allein lebenden älteren Personen. Das stellt zwar in gewisser Weise eine Einschränkung dar, da Mehrpersonenhaushalte von den entwickelten Techniken in der Regel nicht berücksichtigt werden; diese Einschränkung ist jedoch durchaus nachvollziehbar: Bei mehreren Personen im Haushalt kann eher davon ausgegangen werden, dass die jeweils andere Person in der Lage ist zu helfen, bzw. Hilfe zu rufen. Außerdem stellt es bei manchen Szenarien eine nur schwer zu überwindende Hürde dar, mit bestimmten (meist fest installierten) Sensoren detektierte Tätigkeiten dann einer konkreten Person zuzuordnen. So kann es beispielsweise für die Erkennung und Zuordnung von Aktivitäten essentiell sein, welche der anwesenden Personen gerade einen Schalter betätigt hat; das ist aber je nach Vorhandensein von zusätzlichen Sensorinformationen nur sehr schwer, bzw. in der Regel auch gar nicht unterscheidbar.

Projektstruktur

Das Projekt gliederte sich zunächst in acht verschiedene Arbeitspakete, wobei sich die ersten vier Arbeitspakete an konkreten Szenarien orientierten und die folgenden vier Querschnittsthemen behandelten. Die Verzahnung der einzelnen Arbeitspakete ist in Abbildung 1.3 dargestellt. Im Folgenden werden die Aufgaben der für diese Arbeit relevanten Pakete (AP1 bis AP5) kurz dargestellt, da die Szenarien der Arbeitspakete 1 bis 4 alle auf derselben technischen Plattform (AP5) lauffähig sein sollen, welche auch einen Teil der technischen Basis für die Beiträge dieser Arbeit darstellt. Die eigenen Arbeiten werden ab Kapitel 3 gesondert herausgestellt – an dieser Stelle soll zunächst ein allgemeiner Überblick über die technischen Fragestellungen gegeben werden.

[3] http://www.altersgerechte-lebenswelten.de

Neben den angesprochenen technischen Fragestellungen wurden im Gesamtprojekt auch eine Vielzahl weiterer Aufgaben behandelt. So wurden unter anderem auch medizinische, geriatrische, soziologische, ökonomische, ethische und psychologische Problemstellungen betrachtet, welche aber insgesamt zu vielschichtig und zu umfangreich sind, um sie an dieser Stelle angemessen zu würdigen.

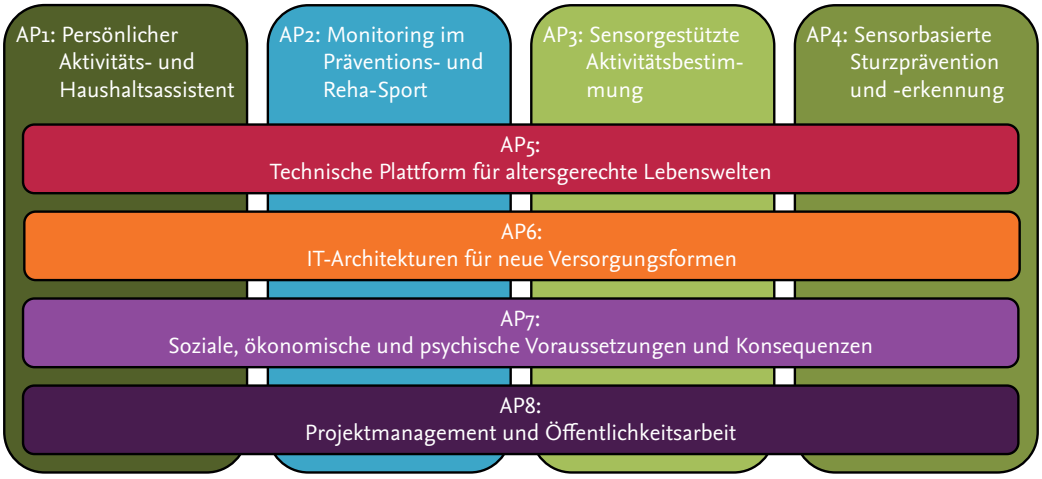


Abbildung 1.3: Die Struktur und die einzelnen Arbeitspakete des GAL-Projekts in den ersten Projektjahren.

1.1.1 Persönlicher Aktivitäts- und Haushaltsassistent

Dieses Arbeitspaket sollte ein Assistenzsystem entwickeln, umsetzen und evaluieren, mit dem in erster Linie Erinnerungen und Empfehlungen für ältere Menschen im Alltag umgesetzt werden können und so deren selbstständiges Leben unterstützt wird. Die Anforderungen sind hierbei unter anderem eine *ambiente*, also wenig auffällige und wenig invasive Umsetzung, die sich einfach in vorhandene Wohnungen integrieren lässt. Die Nutzerinteraktion sollte dabei neben der klassischen Steuerung per Bildschirm, Maus und Tastatur auch über eine Sprachsteuerung geschehen. Die spezifischen Erinnerungen sollten sowohl akustisch als auch optisch ausgegeben werden können und so sowohl seh- als auch hörbehinderte Menschen unterstützen können. So sollten beispielsweise unterschiedliche Lichtfarben einer Zimmerlampe oder unterschiedliche akustische Signale verschiedenen Ereignissen zugeordnet werden können.

Eigene Arbeiten wurden in diesem Arbeitspaket nicht durchgeführt, jedoch sollen sich die Entwicklungen auf der später in Abschnitt 1.1.5 vorgestellten *Technischen Plattform* integrieren lassen.

1.1.2 Monitoring im Präventions- und Reha-Sport

Das zweite Arbeitspaket zielte auf die Überwachung von körpernah aufgenommenen Vitalparamtern ab. Diese Daten, wie beispielsweise Puls oder Atemfrequenz, sollen dabei am Körper einer Person aufgenommen werden – und zwar, während die Person sich in ihrer gewohnten Umgebung befindet. Sinnvoll ist das beispielsweise für zu Hause durchzuführende Sport- und Rehabilitationsübungen, welche dann aus der Ferne überwacht und gegebenenfalls gesteuert werden können.

Zur Aufnahme der Vitalparamter sollte ein *drahtloses körpernahes Sensornetzwerk (Wireless Body Area Network (WBAN))* entwickelt werden, welches die aufgenommenen Daten zunächst per Funk an eine häusliche Basisstation versendet. Von dort sollten die Daten dann wiederum zu entfernten Diensten, wie z.B. der technischen Infrastruktur eines Krankenhauses, übertragen werden. Das Training sollte also auf einem Ergometer in der Wohnung des Patienten stattfinden, wobei die Vitalparameter aus der Ferne überwacht werden können und gegebenenfalls auch aus der Ferne Einfluss auf das Trainingsprogramm genommen werden kann.

Im Lauf des Projekts wurde dieses Arbeitspaket jedoch aufgegeben und die grundlegenden Tätigkeiten auf andere Arbeitspakete (3, 4 und 5) verteilt. Die Fragestellung an sich bleibt jedoch auch für zukünftige Projekte relevant, da in diesem Szenario (im Gegensatz zu vielen *überwachenden* Szenarien) auch *unterstützende* Aspekte berücksichtigt werden.

1.1.3 Sensorgestützte Aktivitätsbestimmung

Welche Sensoren geeignet sind, *Aktivitäten des täglichen Lebens (Activities of daily living (ADL))* zu bestimmen, sollte in diesem Arbeitspaket identifiziert werden. In [3] wurden erste (ernüchternde) Ergebnisse bereits veröffentlicht. Es wurden sowohl stationäre als auch tragbare Sensoren innerhalb von Wohnungen eingesetzt. Ziel war es zunächst, Aktivitäten wie die Nahrungszubereitung oder -aufnahme zuverlässig detektieren zu können. Im späteren Projektverlauf wurde die realistischere Erkennung nächtlicher Toilettengänge angegangen. Als stationäre Sensoren kamen neben diversen Sensoren aus dem Bereich der Hausautomation (Hausbussysteme) hier auch Lichtschranken, Ultraschallsensoren, Bewegungsmelder, Kameras, Mikrophone, vernetzte Elektrogroßgeräte (Kühlschrank, Waschmaschine, Herd) und Laserscanner zum Einsatz. Außerdem wurden tragbare und batteriebetriebene Beschleunigungssensoren eingesetzt, welche – befestigt am Körper des Probanden – Bewegungen erfassen sollten, um so die Aktivität des Tragenden zu bestimmen.

1.1.4 Sensorbasierte Sturzprävention und -erkennung

Eine Sturzerkennung wurde in diesem Arbeitspaket mit diversen Sensoren umgesetzt. Dabei kamen stationäre optische Sensoren (Kameras) sowie fest installierte akustische Sensoren (Mikrophone) zum Einsatz. Außerdem wurden auch hier tragbare Beschleunigungssensoren eingesetzt, welche die abgetastete Beschleunigung per Funk an die Technische Plattform sendeten, auf welcher die übermittelten Rohdaten dann ausgewertet wurden.

Die Sturzprävention hingegen sollte auf einer Sturzrisikoanalyse fußen, welche wiederum auf einer Ganganalyse aufbaut. Die Gangparameter wurden dabei sowohl mit Kameras als auch mit Beschleunigungssensoren bestimmt. Ebenfalls angedacht (aber nicht umgesetzt) war die Verbesserung der Ganganalyse durch zusätzliche tragbare Sensoren, wie beispielsweise Gyroskope, Magnetometer oder Luftdrucksensoren.

1.1.5 Technische Plattform für altersgerechte Lebenswelten

Die Technische Plattform soll laut Projektdefinition die Basis für alle Szenarien und Arbeitspakete darstellen. Laut Spezifikation soll es sich dabei um einen Standard-PC (i386-Architektur) mit einem 32-bit Linux Betriebssystem handeln. Im Projektjargon hat sich der Name *Multi-Services Home Platform (MSHP)* für diese Plattform etabliert; diese Bezeichnung wird auch im weiteren Verlauf verwendet werden. Pro Wohnung, bzw. pro alleine lebender Person ist eine MSHP-Installation vorgesehen. In Kapitel 8 wird detailliert auf die Hard- und Softwareausstattung der MSHP eingegangen.

1.1.6 Datenschutzkonzept

Datenschutz wird gemeinhin als *Schutz der Privatsphäre, Schutz vor missbräuchlicher Verwendung* der Daten oder *informationelle Selbstbestimmung* umschrieben. Das GAL-Projekt hat sich ein ebenso einfaches wie durchaus erwähnenswertes Datenschutzkonzept gegeben:

- Alle Daten gehören der Person, die sie generiert hat, bzw. die über sie aufgezeichnet wurden.
- Dazu bleiben alle Daten auf der jeweiligen MSHP, die dem jeweiligen Benutzer gehört.
- Keine Rohdaten, wie beispielsweise einzelne Messwerte von Sensoren, dürfen die MSHP ohne Einwilligung des Nutzers verlassen.

- Jegliche Berechnungen auf Basis der aufgezeichneten Daten finden nur auf der benutzereigenen MSHP statt.
- Nur (vor-)verarbeitete und aggregierte Meldungen, wie z.B. ein *detektierter Sturz* oder ein erkanntes *erhöhtes Sturzrisiko*, werden nach außen kommuniziert – beispielsweise zu Hausnotrufzentralen oder vorher festgelegten Freunden und Angehörigen.
- Das System darf von außen, also beispielsweise über das Internet, nicht zugänglich sein.

Auf diese relativ strikten Regeln wird in Kapitel 8 noch genauer eingegangen werden.

1.2 Ein Blick in die Nachbarschaft: Ähnliche Projekte

Wie bereits angedeutet, gibt es eine Vielzahl von Projekten, die sich auf unterschiedlichen Ebenen der Herausforderung der älterwerdenden Gesellschaft stellen. Während das Problem der Altersvereinsamung in manchen anderen Kulturkreisen eher unbekannt ist, bleibt der Geburtenrückgang bei gleichzeitiger verlängerter Lebenserwartung eine globale Herausforderung für industrialisierte Gesellschaften. Deshalb sollen in diesem Abschnitt einige exemplarische Projekte, die sich mit verwandten Gesundheitspflege-, Notfall- und/oder AAL-Szenarien befassen, kurz vorgestellt werden. Die schiere Menge macht es an dieser Stelle unmöglich, auf alle Projekte im anvisierten Bereich detailliert einzugehen; die vorgestellte Auswahl wurde daher aufgrund ähnlicher technischer Fragestellungen und verschiedener, für diese Arbeit als relevant erachtete, Teillösungen getroffen.

Im Folgenden werden kurz die Projekte WASP, CodeBlue und SmartAssist vorgestellt, um die Breite bisheriger Arbeiten und des bei diesen noch weiterhin bestehenden Forschungsbedarfs anzudeuten. Weitere Diskussionen von verwandten Arbeiten folgen in den jeweiligen Kapitel mit direktem Bezug zu den dort behandelten Aspekten.

1.2.1 Wirelessly Accessible Sensor Populations (WASP)

Das im August 2010 abgeschlossene EU-FP6-Projekt WASP [4] behandelte den Einsatz von drahtlosen Sensornetzen in zahlreichen unternehmerischen und industriellen Bereichen. Der Fokus lag dabei auf der einfachen Bereitstellung [5] und Integration von drahtlosen Sensornetzen, wobei hierbei der gesamte Bereich von grundlegender Hardware, über Sensoren, Kommunikation, Middleware [6] bis hin zu Anwendungen abgedeckt werden sollte. Eine oft publizierte Anwendung aus diesem Projekt ist die Fernüberwachung von älteren Menschen [7]. Hier gibt es durchaus Ähnlichkeiten zur Aktivitätsüberwachung im GAL Projekt; in ihrem *Elderly Care General Scenario* sollen unterschiedlichste Sensoren zum Einsatz kommen: Zum einen sollen Body Area Network (BAN)-Sensorknoten verwendet werden, um medizinische und Aktivitätsdaten zu sammeln. Zum anderen soll ein sogenanntes *Wireless Ambient Sensor Network* Umgebungsinformationen liefern.

Die Beschreibungen sind teilweise nur auf konzeptionellem Niveau. Während Middleware und Sensorhardware sowohl implementiert als auch evaluiert wurden, trifft das für den Anwendungsfall des *Elderly Care Scenarios* leider nicht zu.

1.2.2 CodeBlue

Im Forschungsprojekt *CodeBlue*- [8][9] der Harvard University wurde ein drahtloses Sensornetz zur Patientenüberwachung entwickelt und in zwei Pilotstudien evaluiert [10]. Der Fokus lag dabei auf den technischen Aspekten, was die Entwicklung von Hardware und Software einschließt. Auf Hardwareseite wurden zunächst drei Erweiterungen für den MicaZ bzw. TelosB-Sensorknoten entwickelt. Es entstanden ein Pulsoxymeter, ein Elektrokardiogramm (EKG) und ein kombiniertes Bewegungserfassungs- und Elektromyografie (EMG)-Modul. Als Betriebssystem auf den Sensorknoten wurde TinyOS eingesetzt.

Das Pulsoxymeter misst die arterielle Sauerstoffsättigung im Blut mit einem an der Fingerspitze fixierten Sensor. Zur Bestimmung des Sauerstoffgehalts wird mit zwei Lichtquellen unterschiedlicher Wellenlänge die Haut durchleuchtet und die Lichtabsorption mit Hilfe von Fotosensoren gemessen; zusätzlich kann darüber auch die Herzfrequenz bestimmt werden.

Ein EKG zeichnet die elektrischen Aktivitäten des Herzens auf – eine genauere Erläuterung der Funktionsweise gibt Abschnitt 4.1.1. Umgesetzt wurde ein 2-Kanal-EKG, welches im Bereich zwischen 100 und 250Hz abtastet und drahtlos übermittelt.

Ebenfalls auf den TelosB-Sensorknoten basiert die Bewegungserfassung. Auf dem hierfür entwickelten Erweiterungsboard wird ein 3-Achsen-Beschleunigungssensor (siehe Abschnitt 4.1.2), welcher mit 100 Hz abgetastet wird, ein 1-Achsen-Gyroskop (siehe Abschnitt 4.1.2), welches mit 100 Hz abgetastet wird, und ein EMG-Modul integriert. Beim EMG wird über zwei auf die Haut zu klebende Elektroden die elektrische Muskelaktivität aufgenommen und mit 1 kHz abgetastet.

Am Ende des Projekts wurde eine eigene Hardwareplattform *miTag* [11] für verschiedene medizinische Sensoren geschaffen, welche jedoch wenig dokumentiert ist. Von Seiten der tragbaren Sensorik gibt es durchaus Schnittmengen mit dem GAL-Projekt – die Anwendungsfälle zielen jedoch klar auf den klinischen Einsatz, während das GAL-Projekt das (personell unüberwachte) häusliche Umfeld fokussiert.

1.2.3 SmartAssist

Das BMBF-geförderte Projekt SmartAssist [12] wurde von 2009 bis 2012 durchgeführt. Ziel war es, ein *sozio-technisches Unterstützungssystem* zunächst für den Raum Lübeck zu schaffen, welches älteren Menschen ermöglicht, länger zu Hause zu leben. Im Gegensatz zum Ansatz des GAL-Projekts, bei welchem möglichst alle Daten innerhalb der Wohnung verarbeitet und verwaltet werden, wird hier auf zentrale Server-Infrastruktur gesetzt, durch welche die Dienste angeboten werden. Ähnlich zum GAL-Projekt ist der Ansatz, Aktivitäten durch in der Wohnung platzierte Sensoren zu bestimmen, wobei hier konsequenterweise *ein* drahtloses Sensornetz die Basis ist und auf aufwändige kabelgebundene Sensorik (wie z.B. Videokameras) verzichtet wird. Mit dem drahtlosen Sensornetz werden unter anderem Wasser- und Stromverbrauch, Temperatur und Luftdruck/-feuchte, das Öffnen und Schließen von Türen und Fenstern sowie der Ort und die Bewegung der Bewohner erfasst.

Als neuer Aspekt wird hier auch ein sogenannter *Market Place* eingeführt, der als eine Schnittstelle zu den Dienstleistungen weiterer Akteure fungiert. Außerdem werden diverse Smartphone-basierte Anwendungsfälle aufgezeigt, die auch ein Vitalparametermonitoring umfassen.

1.3 Wissenschaftliche Beiträge und weiterer Aufbau

Die wissenschaftlichen Beiträge verteilen sich auf unterschiedliche Aspekte eines im Verlauf der Arbeit vorgestellten Gesamtsystems und werden in der folgenden Gliederung jeweils kurz angerissen. Eine Zusammenfassung findet sich am Ende der Arbeit in Kapitel 9.

Nachdem in diesem Kapitel die Motivation und das Umfeld dieser Arbeit umrissen wurden, folgen in den nächsten Kapiteln die konkreten Aufgaben und deren Umsetzungen.

Zunächst werden in Kapitel 2 die Herausforderungen, denen sich diese Arbeit stellt, genauer analysiert: Ausgehend von verschiedenen Szenarien (aus dem GAL-Projekt und aus verwandten Ansätzen), wird ein aggregiertes Szenario entworfen, das die Aspekte *Sturzerkennung, Sturzprävention* (durch Ganganalyse), *Aktivitätserkennung, Vitalparamterüberwachung* und *Hausnotruffunktionalitäten* umfasst. Zusätzlich sollen dabei auch Aspekte der Sicherheit (*security & safety*) und des *Datenschutzes* abgedeckt werden. Das für diese Arbeit entworfene Szenario mündet dann in einer Anforderungsdefinition, die diverse Teilaspekte auf der gesamten Strecke von der Aufnahme der Daten mit körpernaher Sensorik bis hin zu den Backensystemen umfasst.

Aus dieser Anforderungsdefinition folgt in Kapitel 3 die Skizze eines Systems, das diesen Anforderungen

gerecht werden soll. Daneben werden hier bereits für einige Teilaspekte mögliche Umsetzungen durch die Betrachtung verwandter Arbeiten und Systeme in Augenschein genommen. Am Ende des dritten Kapitels erfolgt eine Aufteilung in konkrete Teilaufgaben, die im weiteren Verlauf behandelt werden.

Da der Fokus auf der Erfassung, dem Verarbeiten und dem Übertragen von körpernah aufgenommenen Daten liegt, werden jedoch zunächst in Kapitel 4 die zu erwartenden Datenarten und -raten im betrachteten Anwendungsgebiet genauer analysiert. Neben der Betrachtung unterschiedlicher aufzunehmender Parameter wird hier auch auf die erzeugten Datenraten von einzelnen (für den weiteren Verlauf relevanten) Sensoren eingegangen sowie mögliche Übertragungstechniken für diese Daten diskutiert.

Da sich bei der Betrachtung der für den Systementwurf relevanten tragbaren Sensorsysteme in Kapitel 3 gezeigt hat, dass von den vorhandenen und verfügbaren Systemen keines in der Lage ist, alle gestellten Anforderungen zu erfüllen, wird in Kapitel 5 ein tragbares Sensorsystem entworfen, implementiert und evaluiert, das auf der einen Seite in der Lage ist, durch die Sensor- und Speicherausstattung die Anwendungsfälle zu unterstützen, auf der anderen Seite aber auch universell einsetzbar ist.

Mit diesem Sensorsystem, dem INGA-Sensorknoten, werden in Kapitel 6 verschiedene Aspekte der Datenübertragung von der körpernahen Sensorik zu einer Basisstation behandelt. Dabei wird zunächst in Abschnitt 6.1 der Nutzen und die Umsetzbarkeit von unterbrechungstoleranten Netzen (Disruption Tolerant Networks (DTNs)) für die Szenarien theoretisch analysiert. Anschließend wird μDTN (μDTN), ein unterbrechungstolerantes Kommunikationsprotokoll für Wireless Sensor Networks (WSNs), vorgestellt und für die Anwendungsfälle angepasst und in diesen evaluiert. Da das Protokoll eine implizite Synchronisierung der im Unterbrechungsfall aufgenommenen Daten vornimmt, können mit dem daraus konzipierten System mehrere Anwendungsfälle gleichzeitig bedient werden, die sonst unterschiedliche Übertragungsmechanismen oder gar unterschiedliche Hardwaresysteme voraussetzen.

Zur Absicherung des Funkkanals zwischen mobiler Sensorik und Basisstation werden in Abschnitt 6.2 zunächst verschiedene Verfahren und Systeme zur verschlüsselten Datenübertragung in WSNs vorgestellt. In diesem Rahmen wird auch eine AES-Implementierung für INGA umgesetzt und evaluiert. Anschließend wird ein System für eine mit One-Time-Pads (OTPs) gesicherte Verschlüsselung entworfen, implementiert und evaluiert, welches auch die Schlüsselerzeugung und -verteilung umfasst.

Für den Fall, dass die Datenrate des Funkkanals oder die Größe des zur Verfügung stehenden Speichers nicht ausreicht, um die anfallenden Daten zu verarbeiten, werden in Abschnitt 6.3 zunächst verschiedene Möglichkeiten zur verlustfreien Datenreduktion vorgestellt. Nach theoretischen Betrachtungen und Evaluationen (mithilfe von im realen Umfeld aufgenommenen Datensätzen aller zur Verfügung stehenden Sensoren) verschiedener Reduktionsmechanismen wurde mit der „angepassten Deltakodierung“ ein für den Anwendungsfall passendes Verfahren gefunden, das (im Anwendungsfall) in der Lage ist, die anfallenden Daten mit wenig Rechenaufwand platzsparender zu kodieren als die eigentlich optimale Huffman-Kodierung.

Um auch außerhalb des Kommunikationsradius der Basisstation Notfallmeldungen absetzen zu können, wird in Kapitel 7 ein System (bestehend aus einem Smartphone und einem (um ein Bluetooth-Interface erweiterten) Sensorknoten) skizziert, implementiert und evaluiert, das in der Lage ist, auch unterwegs über das Mobilfunknetz Alarmmeldungen zu versenden und dabei möglichst energiesparend funktioniert.

In Kapitel 8 werden verschiedene Herausforderungen, die an die Infrastrukturelemente (häusliche Basisstation und Backend) gestellt werden, behandelt. Um einem Datenverlust vorzubeugen werden hier verschiedene Techniken zur Datensicherung, zum Monitoring und zur Fernwartung über ein unsicheres Medium wie das Internet diskutiert, umgesetzt und evaluiert. Die gefundene Lösung, die auf einem durch das häusliche Gateway initiiertem VPN-Tunnel basiert, wahrt dabei die am Anfang dieses Kapitels aufgestellten Grundsätze zum Datenschutz (Abschnitt 1.1.6).

Kapitel 9 schließt die Arbeit mit einem Ausblick ab.

2 Herausforderungen

„The greatest challenge to any thinker is
stating the problem in a way that will allow a solution“

Bertrand Russell

In diesem Kapitel sollen zunächst verschiedene Herausforderungen aufgezeigt und daraus die Problemstellung für diese Arbeit abgeleitet werden. Die Grundlagen bilden dafür zunächst die Szenarien des GAL-Projekts. Kombiniert mit anderen Szenarien aus verwandten Projekten, dem allgemeinen medizinischen-, bzw. notfallmedizinischen und dem AAL-Bereich wird ein umfassendes Szenario entworfen, welches alle Aufgabenbereiche und Fragestellungen dieser Arbeit umfasst. Im Anschluss werden daraus konkrete Subszenarien für die spezifischen Herausforderungen definiert.

Allen im Folgenden erläuterten Szenarien ist gemein, dass Daten am menschlichen Körper aufgenommen werden, sodass die übergreifende Fragestellung für diese Arbeit lautet:

> **Wie können Daten sicher und zuverlässig vom menschlichen Körper zu einer entfernten verarbeitenden Einheit übertragen werden, während sich der Mensch frei im Raum bewegt?**

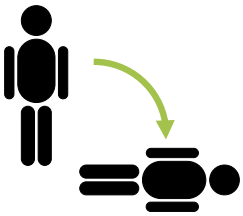
Aus diesem Grund werden die GAL-spezifischen Szenarien auch nur soweit beschrieben, als dass sie tragbare und körpernahe Sensoren beinhalten – andere, in GAL durchaus intensiv behandelte, fest installierte Sensoren würden an dieser Stelle den Fokus zu sehr verwischen. Beispielsweise sind durchaus optische Sensoren (Videokameras) [13], Arrays von Infrarotsensoren [14], Laser Scanner [15] oder sogar Vibrationssensoren im Fußboden [16] zur Sturzdetektion sinnvoll nutzbar – tragbar und drahtlos sind sie jedoch nicht. Auch andere Sensoren, wie Bewegungsmelder zur Positionsbestimmung, Lichtschranken und Sensoren der Hausautomation bleiben bei dieser Festlegung der Szenarien deshalb (zunächst) unberücksichtigt. Unabhängig davon lassen sich natürlich auch fest installierte und mit Kabeln angebundene Sensoren durch drahtlose und (eigentlich) mobile Knoten ersetzen, was aber die Fragestellung nicht erweitern würde.

In [17] werden einige Anwendungsbereiche für WSNs im Gesundheitsbereich vorgestellt. Dabei wird dort zunächst grundsätzlich zwischen der *Überwachung von Vitalparametern* (Physiological Monitoring) und der *Bewegungs- und Aktivitätserkennung* (Motion and Activity Monitoring) unterschieden. Als Endanwendung wird dort der Einsatz in größeren *Studien zur Erforschung von Physiologie und Verhalten* (Large-Scale Physiological and Behavioral Studies) mit Hilfe von WSNs angeführt. Alle diese Bereiche werden auch im Folgenden betrachtet und wurden teilweise im Rahmen des GAL-Projekts behandelt.

2.1 Szenarien im GAL-Projekt

Die für eine am Körper getragene Sensorik relevanten Szenarien entstammen den GAL-Arbeitspaketen 2 bis 4. Konkret handelt es sich dabei um die *Sturzerkennung*, *Sturzprävention* und die *Aktivitätserkennung*. Die im aufgegebenen Arbeitspaket 2 angegangene (wenn auch nicht umgesetzte) *Fernüberwachung beim Rehabilitationssport* dient als weitere Anregung, da dort unter anderem geplant war, Vitalparameter des menschlichen Körpers aufzunehmen, was auch im allgemeinen medizinischen Bereich von besonderer Bedeutung ist, weswegen dieser Punkt im nächsten Abschnitt gesondert behandelt wird.

2.1.1 Sturzerkennung



Wie der Name vermuten lässt, ist es das Ziel der Sturzerkennung, den – ansonsten unbeobachteten – Sturz einer Person automatisch und zuverlässig zu detektieren. Dazu werden in der Literatur unterschiedliche Ansätze mit diversen Algorithmen [18], Sensoren und Systemen behandelt [19]. In der praktischen Anwendung sind solche Sturzerkennungssysteme mit automatischen und autonomen Alarmierungssystemen gekoppelt, welche einen detektierten aufgetretenen Sturz dann an Notfalleinsatzkräfte oder Hausnotrufzentralen weiterleiten und so im Idealfall für eine schnelle Hilfe sorgen.

Für eine Sturzerkennung eignen sich unterschiedliche Sensoren und die meisten Publikationen setzen dabei auf tragbare und batteriebetriebene Sensoren, wobei auch die Position dieser Sensoren am Körper ein Teil gegenwärtiger Forschung [20] ist.

In den meisten Fällen wird dabei die Beschleunigung – also die Änderung der Geschwindigkeit – eines Körpers gemessen [21]. Dazu kommen in der Regel Beschleunigungssensoren *(Accelerometer)* zum Einsatz, die auf mikromechanischen Elementen (Microelectromechanical systemss (MEMSs)) basieren.

In letzter Zeit werden auch häufiger gyroskopische Sensoren – also Sensoren, die eine Veränderung der Lage im Raum detektieren können – zur Sturzerkennung eingesetzt [22]. Diese Sensoren sind meist auch in MEMS-Technologie ausgeführt. Auch die Kombination von Accelerometern und Gyroskopen [23] ist anzutreffen.

Außerdem gibt es Versuche mit am Körper getragenen Luftdrucksensoren [24]. Hier macht man sich zu nutze, dass sich der vom Sensor gemessene Luftdruck im Falle eines Sturzes kurzfristig ändert.

Eine Sturzerkennung ergibt nur dann Sinn, wenn zeitnah auf einen detektierten Sturz reagiert werden kann. Es ist also eine sofortige *online* Auswertung der Daten notwendig. Außerdem ist eine (ständige) Funkverbindung zu einer Basisstation, welche Notrufe absetzen kann, sinnvoll. Wo jedoch die Algorithmen zur Sturzerkennung ausgeführt werden – ob auf der Basisstation oder auf dem tragbaren Gerät – ist von Implementierung zu Implementierung verschieden. Im GAL-Projekt wurden die Algorithmen zur Sturzerkennung auf der Basisstation (MSHP) ausgeführt – von der tragbaren Sensorik wurden dorthin lediglich die auszuwertenden Messwerte per Funk übertragen. Dass eine Sturzerkennung auch auf mobilen Geräten umsetzbar ist, wird zum Beispiel in [25] und [26] beschrieben.

2.1.2 Sturzprävention

Ein häufig eingesetztes Instrument zur Bestimmung des Sturzrisikos ist der sogenannte *Timed Up & Go*-Test [27]. Dieser Test findet in der Regel unter ärztlicher Aufsicht statt: Der Proband sitzt zu Beginn des Tests auf einem Stuhl und wird aufgefordert, aufzustehen, eine bestimmte Strecke zu gehen, sich umzudrehen, zurückzugehen und sich wieder hinzusetzen. Die beaufsichtigende Person misst die Zeit, die vom Probanden für die Prozedur benötigt wird und anhand dieser gemessenen Zeit kann das Sturzrisiko prädiziert werden, wobei längere Zeiten für ein höheres Sturzrisiko sprechen. In [28] wurde dieser Test automatisch unter Zuhilfenahme von im Fußboden eingebauten Sensoren (Sensorflur) umgesetzt.

Eine Sturzprävention soll im GAL-Projekt durch eine *Ganganalyse* stattfinden. Die vereinfachte Idee dahinter ist, dass eine Veränderung, bzw. Verschlechterung der Gangparameter eine Aussage über ein verändertes Sturzrisiko erlauben. Gangparameter sind beispielsweise die Schrittlänge, die Geschwindigkeit (also die Zeit, welche für einen bestimmten zurückgelegten Weg benötigt wird) oder auch die Regelmäßigkeit, mit der Schritte stattfinden.

Als tragbare Sensoren kommen dabei neben Accelerometern [29] auch vermehrt Gyroskope zum Einsatz [30][31]. Eine Kombination dieser beiden Sensorarten verspricht eine verbesserte Detektionsgenauig-

keit [32].

Da die Sturzprävention auf eine längere Beobachtung abzielt und auch Vergleiche mit Messungen vergangener Tage oder Wochen berücksichtigt werden müssen, ist eine sofortige Auswertung der Daten nicht zwingend. Es kann durchaus sinnvoll sein, zunächst Daten über einen längeren Zeitraum aufzunehmen und sie nach Vollendung der Messreihe *offline* auszuwerten. Wenn diesen Daten sofort verfügbar wären, wäre das für ein solches System jedoch auch kein Nachteil.

2.1.3 Aktivitätserkennung

Bei der *Aktivitätserkennung* ist zunächst zu unterscheiden, ob *Aktivitäten des täglichen Lebens* (ADL) erkannt werden sollen, oder – etwas abstrakter – *allgemeine Aktivität*. Mit am Körper getragenen Accelerometern und Gyroskopen lassen sich einfache „Tätigkeiten" gut detektieren. So kann *sitzen* von *stehen*, *liegen* und *laufen* einigermaßen zuverlässig unterschieden werden [33]. Bei wirklichen zu detektierenden Aktivitäten des täglichen Lebens, wie beispielsweise die *Nahrungszubereitung*, die *Nahrungsaufnahme* oder einfach *Fernsehen*, liefern die Bewegungssensoren schlicht zu wenig Informationen. So ist es offensichtlich, dass ohne zusätzliche Kenntnisse der Umgebung das *Sitzen* nicht zwangsweise vom *Fernsehen* unterschieden werden kann, können doch diese beiden Tätigkeiten sogar gleichzeitig ausgeführt werden. Auch das Wissen über einen eingeschalteten Fernseher und eine sitzende Person im selben Raum kann nicht zwangsläufig zur Tätigkeit *Fernsehen* zusammengefasst werden, kann doch die Person auch gleichzeitig telefonieren oder schlafen.

Dass überhaupt eine körperliche Aktivität stattfindet, lässt sich jedoch relativ einfach bestimmen [34], wobei auch hier die Position der Sensoren am Körper eine wichtige Rolle spielt [35].

Als Aktivitätssensoren werden in der Regel auch MEMS-basierte Accelerometer und Gyroskope verwendet. Aber auch hochpräzise Luftdrucksensoren liefern relevante Informationen [36]: Da sich der Luftdruck mit der Höhe ändert, kann die zusätzliche Information über den sich ändernden Luftdruck beispielsweise Auskunft darüber geben, ob gerade eine Treppe hinauf- oder hinabgegangen wird [37] oder auf welchem Stockwerk sich eine Person befindet [38].

Wird die Erkennung von Aktivitäten auf den häuslichen Bereich beschränkt, entgehen einem solchen System gerade bei noch mobilen Menschen wichtige Tätigkeiten des täglichen Lebens. Es wäre demnach wünschenswert, auch die Aktivitäten quantifizieren zu können, die außerhalb der Wohnung stattfinden. Da die Messwerte der Aktivitätserkennung eher Richtwerte für einen längeren Zeitraum sind, ist eine sofortige *online* Auswertung in der Regel nicht notwendig und eine *offline*-Analyse der Daten völlig ausreichend.

2.2 Weitere Szenarien im medizinischen und AAL-Bereich

Auch in anderen Bereichen des umgebungsunterstützten Lebens, der Medizin oder aber auch im Bereich der Notfallversorgung gibt es Szenarien, in denen tragbare und drahtlose Sensorsysteme zum Einsatz kommen.

2.2.1 Vitalparameterüberwachung

Ursprünglich war die Aufnahme von Vitalparametern auch im GAL-Projekt vorgesehen, um dort eine Trainingsüberwachung aus der Ferne zu ermöglichen. Aber auch im allgemeinen medizinischen Bereich, ist das Wissen über die Vitalparamter eines Menschen von großer Bedeutung. Mittlerweile wächst besonders bei Sportlern der Trend, sich selbst zu quantifizieren, also alle erdenklichen Körperdaten möglichst umfassend zu erfassen und kontinuierlich aufzuzeichnen, um sie anschließend auswerten zu können und so beispielsweise Trainingsmethoden zu verbessern [39]. Je nachdem, ob man sich im professionellen medizinischen oder im semi-professionellen sportlichen Bereich bewegt, kommen unterschiedliche Sensorsysteme zum Einsatz. Während das Langzeit-EKG als medizinisches Diagnoseinstrument angewendet wird, um über

einen längeren Zeitraum EKG-Daten zu sammeln und anschließend auszuwerten, kommen beim Sport eher Pulsraten-Messgeräte zum Einsatz, wobei aber auch hier ein Trend zu aufwändigerer Sensorik zu erkennen ist.

Je nach Anwendungsfall müssen die Vitalparameter entweder sofort online zur Verfügung stehen oder aber für eine spätere Auswertung gespeichert werden. So werden die gespeicherten Daten eines Langzeit-EKG in der Regel erst nach dem Abbau der Sensoren ausgewertet, während bei der Fernüberwachung und Steuerung von Trainings auf Echtzeitdaten nicht verzichtet werden kann. Im Freizeitsport verlangen die ambitionierten Sportler von ihren Pulsmessgeräten sogar beides auf einmal: Zum einen soll der aktuelle Puls direkt angezeigt werden, zum anderen aber auch zusammen mit beispielsweise Positionsdaten zur späteren Auswertung gespeichert werden

2.2.2 Hausnotruf

Das was in Deutschland unter dem Namen *Hausnotrufsystem* vermarktet wird, ist international als Personal Emergency Response System (PERS) bekannt und gehört schon seit längerem zum Dienstangebot der meisten Rettungs- und Pflegedienste [40]. Es handelt sich dabei in der Regel um eine Kombination aus einem tragbaren Gerät und einer an das Telefonnetz angeschlossenen Basisstation. Das tragbare Gerät wird als Kette um den Hals oder als Armband um das Handgelenk getragen und verfügt in der Regel nur über einen Taster, mit dem der Träger des Geräts Alarm auslösen kann.

Der vorgesehene Einsatzzweck ist also, dass eine mit einem Hausnotrufsystem ausgestattete Person, sobald sie sich in einer Notlage befindet, aus der sie sich nicht selbständig wieder befreien kann (beispielsweise ein Sturz), den Taster betätigt. Dieses Ereignis wird dann per Funk an die Basisstation übermittelt, welche dann wiederum automatisch die entsprechende Rettungsleitstelle oder den jeweiligen Dienstleister über das Telefonnetz kontaktiert. In der realen Anwendung lassen sich jedoch verschiedenste Verhaltensweisen von (vermeintlichen) Nutzern eines Hausnotrufsystems ausmachen [41]:

Normale Benutzung: Der angestrebte Anwendungsfall ist, dass die Person das tragbare Gerät tagsüber am Körper trägt und den Taster benutzt, um im Notfall Alarm auszulösen.

Nichtbenutzung: Oft kommt es vor, dass, wenn ein vom Hausnotrufdienstleister beauftragter Servicetechniker die Batterien des tragbaren Geräts wechseln soll, das Gerät zunächst lange in Schubladen gesucht werden muss, da es offensichtlich nicht genutzt wird. Dieser Fall tritt häufig auf, wenn Angehörige ein solches System verschenkt haben, der potentielle Nutzer aber von dem Nutzen nicht überzeugt ist.

Verweigerte Nutzung: Selbst wenn das Gerät am Körper getragen wird und wenn dann eine Notfall-/Unfallsituation eintritt, aus welcher sich das Opfer nicht selbständig befreien kann, wird in einigen Fällen die Benutzung des Tasters verweigert. Begründet wird das oft damit, dass man „niemandem zur Last fallen“ wollte und lieber (hilflos) auf den ohnehin regelmäßig kommenden Pflegedienst warten wollte.

Überängstliche Nutzung: Sehr ängstliche oder verwirrte Menschen können dazu tendieren, das tragbare Gerät nicht nur tagsüber sondern auch während des Schlafens zu tragen. Wenn die Person sich dann im Schlaf auf den Taster rollt oder ihn im Traum unbeabsichtigt bedient, kommt es vermehrt zu Fehlalarmen.

Vermeidungsverhalten: Darüber hinaus gibt es Menschen, die sich zwar mit einem Hausnotrufsystem sicher fühlen, aber sich dessen bewusst sind, dass dieses nur innerhalb der eigenen Wohnung funktioniert. Ein Verlassen der Wohnung wird vermieden, da dort das System nicht mehr für gewohnte oder wahrgenommene Sicherheit sorgen kann.

Während man der *Nichtnutzung* eines solchen Systems nur mit stationärer Sensorik (beispielsweise Kameras, Bewegungsmelder, etc.) begegnen kann, was wiederum nicht im Fokus dieser Arbeit steht, lassen sich die anderen Fälle sehr wohl auch mit tragbarer Sensorik abhandeln. Die *verweigerte Nutzung* kann zum Beispiel mit einer automatischen Sturzerkennung (s.o.) „übergangen" werden. In diesem Fall kann ein solches System auch ohne Nutzerinteraktion einen Alarm auslösen. Um die *überängstliche Nutzung* (und damit die Fehlalarme) zu reduzieren, könnte man auf ein System setzen, welches nur noch autonom und automatisch reagiert und eine Nutzerinteraktion gar nicht erst zulässt; das würde aber wahrscheinlich zu einer verminderten Akzeptanz führen. Ein System, welches sowohl in der Wohnung als auch überall anders funktioniert, kann helfen, ein eventuelles *Vermeidungsverhalten* abzubauen und so dem Benutzer auch außerhalb der Wohnung die gewünschte Sicherheit bieten.

Zusätzlich zum aktiven, durch den Nutzer ausgelösten, Alarm wird oft eine passive Alarmierung eingesetzt, die vom Prinzip her einer *Totmanneinrichtung* entspricht: Der Benutzer muss regelmäßig (beispielsweise ein Mal pro Tag) eine Taste auf der Basisstation drücken; so signalisiert die Person, dass „alles in Ordnung" ist. Bleibt das passive Signal aus, wird nach einer bestimmten Zeit von Seiten des entsprechenden Dienstleisters angerufen oder direkt Hilfe vorbei geschickt.

Da ein Hausnotrufsystem auf Notfälle reagieren muss, müssen auch die Ereignisse sofort *online* ausgewertet und behandelt werden; eine verzögerte Auswertung wäre fatal.

2.3 Abgeleitetes allgemeines Szenario

Aus diesen fünf konkreten Szenarien soll nun ein Gesamtszenario entwickelt werden, welches die interessanten Aspekte der vorgestellten Szenarien vereint. Allen Szenarien ist die Aufnahme von Messwerten durch tragbare Sensorik am menschlichen Körper gemein. Teilweise ist eine sofortige Datenauswertung essentiell, teilweise ist eine dauerhafte Aufzeichnung der Daten unerlässlich. Bei manchen Szenarien ist die Funktionalität auch außerhalb der Wohnung wünschenswert, bei anderen ist sie unerlässlich.

In Tabelle 2.1 wird deshalb die Bewertung der unterschiedlichen Szenarien im Hinblick auf ihre Anforderung an eine sofortige (online) oder spätere (offline) Datenübermittlung aufgezeigt. Außerdem ist hier auch angedeutet, ob eine Funktionalität auch außerhalb der Wohnung nötig oder sinnvoll ist.

Tabelle 2.1: Datenübertragung und Verarbeitung in den Szenarien (+ essentiell / - nicht nötig / o möglich / ∗ wünschenswert).

Szenario	online	offline	drinnen	draußen
Sturzerkennung	+	-	+	∗
Sturzprävention	-	+	+	∗
Aktivitätserkennung	o	o	o	∗
Vitalparameter: Rehasport	+	o	+	∗
Vitalparameter: Freizeitsport	-[1]	+	o	+
Vitalparameter: Langzeit-EKG	-	+	+	+
Hausnotruf	+	-	+	∗

2.3.1 Online vs. Offline Überwachung

Am Beispiel eines EKG soll an dieser Stelle zunächst kurz der Unterschied zwischen dem online-Monitoring mit sofort verfügbaren Daten und dem offline-Monitoring mit verzögert verfügbaren Daten deutlich gemacht werden. In Abbildung 2.1 sind dazu beide Fälle grafisch dargestellt. Im oberen Fall (a) werden alle Daten auf

[1] Auch die im Freizeitsport eingesetzten Geräte, analysieren die Daten nicht online - hier wird lediglich der Puls visualisiert.

einer SD-Karte aufgezeichnet. Diese SD-Karte kann nach erfolgter Messung entnommen und dann an einem Rechner ausgewertet werden. Diese Methode ist äußerst robust, da in der Regel keine Daten verloren gehen und die Festspeicher beispielsweise auch gegen Erschütterungen immun sind. Auf der anderen Seite ist diese Methode auch zeitverzögert, da auf die Vollendung der Messung und gegebenenfalls auf das Eintreffen der SD-Karte beim auswertenden Arzt gewartet werden muss.

Im unteren Fall (b) erfolgt hingegen eine direkte Übermittlung der aufgezeichneten Daten über eine Funkschnittstelle. Das hat den unbestreitbaren Vorteil, dass die Daten sofort zur Auswertung verfügbar sind und so auch direkt auf die Daten reagiert werden kann. Wenn in diesem Fall beispielsweise ein Kammerflimmern detektiert wird, kann der überwachende Arzt rettend eingreifen. Der Nachteil der direkten Übermittlung ist die Fehleranfälligkeit: Solange der sendende Sensor in Funkreichweite der Basisstation ist und der Funkkanal einigermaßen störungsfrei ist, wird ein solches System funktionieren; verlässt aber der Proband den Funkbereich der Basisstation, gehen unweigerlich Daten verloren. Für ein Langzeit-EKG wäre so etwas untauglich, da hierbei ja Daten über einen längeren Zeitraum – und zwar ohne Unterbrechungen – aufgezeichnet werden sollen.

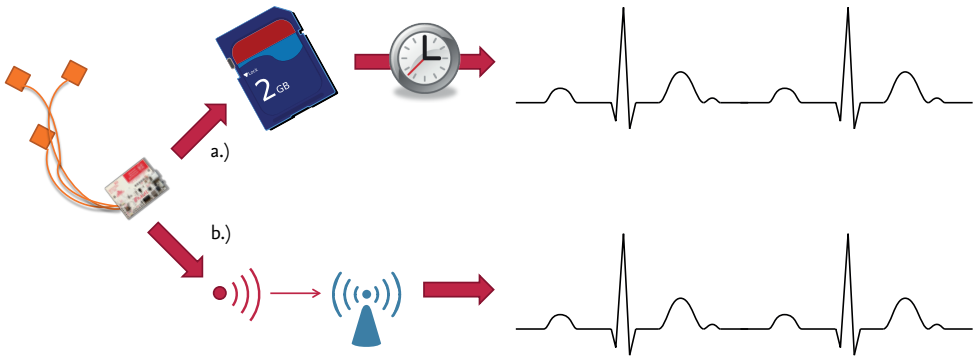


Abbildung 2.1: Datenaufzeichnung (*offline*) und direkte Datenübermittlung (*online*): Bei der offline-Variante (a) werden die Daten zunächst gespeichert und später ausgelesen und ausgewertet. Bei der online-Variante werden die Daten direkt versendet und stehen an der Basis sofort zur Verfügung.

So können schon für das EKG unterschiedliche Ziele identifiziert werden. Im Falle einer Langzeitüberwachung ist die kontinuierliche Speicherung wichtig; wenn eine schnelle Intervention bei der Überwachung im Reha-Sport gegeben sein muss, ist eine online-Übertragung unerlässlich.

Bei der *Aktivitätsüberwachung* verhält es sich ganz ähnlich: Will man zeitnah auf das Gemessene eingehen, braucht man die aufgezeichneten Daten umgehend und in diesem Fall ist eine direkte Übermittlung der Daten zur auswertenden Einheit wichtig. Will man nur eine allgemeine Aktivität feststellen, beispielsweise wie viel sich ein Mensch im Laufe eines Tages bewegt, würde man eher zur robusteren aber verzögerten offline-Auswertung tendieren.

Bei der *Sturzerkennung* und im *Hausnotruf* ist es wiederum eindeutig: Hier ist die sofortige Verfügbarkeit der (ausgewerteten) Daten essentiell, eine Speicherung und spätere Auswertung würde dem Notfallcharakter der zu detektierenden Ereignisse widersprechen.

Die *Sturzprävention* arbeitet ohnehin immer im Vergleich zu zuvor aufgezeichneten Daten, um eine Veränderung feststellen zu können. Da es sich hierbei nicht um eine Notfallintervention handelt, ist es eher wünschenswert einen Datensatz zu späterer Zeit ohne Unterbrechungen zu haben, als einen unvollständigen Datensatz sofort zur Verfügung zu haben.

2.3.2 Funktionalität außerhalb der Wohnung

Ebenfalls in Tabelle 2.1 sind die Notwendigkeiten und Möglichkeiten der einzelnen Szenarien, auch außerhalb der Wohnung zu funktionieren, abgebildet.

Im *Freizeitsport* ist es ja quasi die inhärente Aufgabe, Vitalparameter während des Sports, welcher in der Regel nicht in der Wohnung stattfindet, aufzuzeichnen. Insofern ist hier die Funktionalität außerhalb der Wohnung ein Muss. Ähnlich verhält es sich beim *Langzeit-EKG* – hier sollen ja lange Zeiträume ohne Unterbrechung aufgezeichnet werden, was selbstverständlich auch die Zeiten, in welchen sich der Proband nicht in der Wohnung befindet, mit einschließen muss.

Bei allen anderen Szenarien wäre es jedoch zumindest wünschenswert, wenn diese auch außerhalb der Wohnung funktionieren würden. Beim *Hausnotruf* könnte man so dem in Abschnitt 2.2.2 erwähnten Vermeidungsverhalten entgegenwirken, welches ängstliche Personen eventuell daran hindert die vermeintlich sichere Wohnung zu verlassen. Gleiches gilt für die *Sturzerkennung*, die man auch mit besagtem Hausnotrufsystem koppeln könnte und so eine autonome Funktionalität des Hausnotrufsystems schaffen würde.

Wenn es beispielsweise um die Bewertung der allgemeinen Mobilität geht und eine *Aktivitätserkennung* nur die Aktivitäten wahrnehmen würde, die innerhalb einer Wohnung stattfinden, könnten verzerrte Bewertungen entstehen: Ein Mensch, der täglich mehrere Stunden draußen mit dem Hund spazieren geht und anschließend erschöpft auf dem Sofa fernsieht, würde in seiner Aktivität geringer bewertet als ein anderer Mensch, der das Haus nicht verlässt, aber dafür ein wenig in der Wohnung hin- und herläuft. Dieser Schieflage lässt sich am besten mit einer lückenlosen Aktivitätsüberwachung beikommen, die auch die Aktivitäten außerhalb der Wohnung berücksichtigt.

Ähnliches gilt auch für die Ganganalyse, die zur *Sturzprävention* durchgeführt werden soll: Dass sich Personen in ihrer gewohnten Umgebung sicherer bewegen als in unbekannterer Umgebung ist schnell einzusehen. Warum sich bei der Ganganalyse nur auf die „sicheren" Bereiche verlassen und nicht auch die unbekannten und damit vermeintlich unsicheren Bereich mit einschließen?

Des Weiteren wäre es zumindest eine Option, wenn der *Reha-Sport* nicht nur am heimischen Ergometer durchgeführt werden kann, sondern auf einem richtigen Fahrrad in der Natur stattfinden könnte.

2.3.3 Aggregiertes Szenario

Als aggregiertes Szenario ergibt sich also eine Kombination der Anforderungen der oben beschriebenen Szenarien, was in Abbildung 2.2 symbolisch darstellt ist. Das heißt, ein zu entwickelndes Gesamtsystem soll die Anwendungsfälle *Sturzerkennung*, *Sturzprävention* (Ganganalyse), *Aktivitätserkennung*, *Vitalparameter-Monitoring* und *Hausnotruf* ermöglichen – und zwar sowohl in der Wohnung als auch außerhalb der Wohnung und es soll je nach Anforderung an den spezifischen Anwendungsfall eine direkte Datenübertragung stattfinden können, sowie eine Aufzeichnung der Daten zur späteren Auswertung. Außerdem ist für die Sturzerkennung und den Hausnotruf ein Notfallkanal zur Alarmierung von Hilfskräften vorzusehen. Idealerweise werden alle Szenarien von ein und demselben tragbaren Gerät unterstützt, welches dann je nach Anwendungsfall konfiguriert werden kann. Optional ist hier die direkte Notfallmeldung von diesem Gerät aus, was für eine Alarmmeldung bei Notfällen außerhalb der Funkreichweite der Basisstation wünschenswert wäre.

Erklärtes Ziel ist es also, alle Anwendungsfälle auf demselben Hardware-Software-System zu unterstützen. Wenn mehrere Anwendungsfälle parallel unterstützt werden sollen, müssen in einem solchen System die Messwerte nicht mehrfach erhoben werden, sondern können von mehreren Anwendungen genutzt werden. Als weitere Herausforderungen ergeben sich aus diesem kombinierten Einsatz der Wechsel zwischen online- und offline-Überwachung (Handover / Synchronisation). Außerdem ist die Umsetzung des optionalen Notfallkanals für eine direkte Alarmmeldung (sofern die überwachte Person nicht im Empfangsradius der Basisstation ist) wünschenswert.

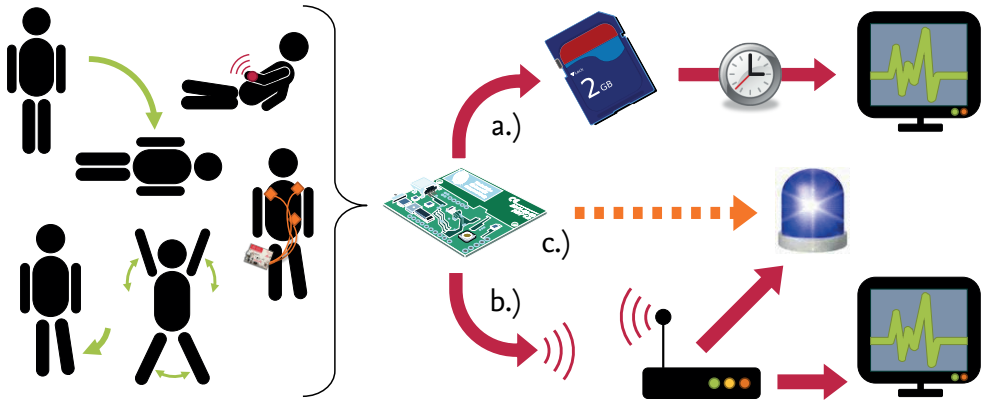


Abbildung 2.2: Aggregiertes Szenario mit einem zentralen tragbaren Gerät für alle Anwendungsfälle. Unterstützung von Speicherung und offline-Auswertung (a), direkter Funkübertragung und Onlineauswertung (b) (mit Notfallmeldung) und optionaler direkter Notfallmeldung (c).

2.4 Anforderungen an die Sicherheit

Neben den rein funktionalen Anforderungen gibt es auch etliche weitere Anforderungen, die ein solches integriertes System erfüllen muss, um letztendlich auch eingesetzt werden zu können. Neben der Einhaltung von Gesetzen und Standards betrifft das sicherlich auch Fragen der Akzeptanz, aber allen voran Fragen der Sicherheit, wobei davon ausgegangen werden kann, dass „höhere Sicherheit“ – was das ist, wird im Folgenden geklärt – auch zu einer größeren Akzeptanz führen wird. Der deutsche Oberbegriff *Sicherheit* kann dabei je nach Standpunkt und Betrachtungsweise etwas anderes umfassen.

Im Englischen unterscheidet man gemeinhin zwischen *security* und *safety*. Ersteres beschreibt den Schutz vor Angriffen von außen – im Folgenden wird dafür der Begriff *Angriffssicherheit* verwendet. Letzteres beschreibt den Schutz vor Schäden, die durch den Betrieb des Systems selbst entstehen können, also die *Betriebssicherheit*. Beide Ziele sind für ein universelles medizinisches oder AAL-System unabdingbar.

Angriffssicherheit: Das System darf von außen möglichst nicht angreifbar sein. Angriffe, die auf das System abzielen, dürfen niemals in der Lage sein, den benutzenden Menschen in irgendeiner Art und Weise zu schädigen. Das umfasst auch den Schutz vor Manipulationen: Wenn Messwerte verfälscht werden, kann dies unter Umständen zu fehlerhaften Diagnosen führen, welche dann in falschen Behandlungen münden.

Betriebssicherheit: Das System an sich sollte keinen Schaden anrichten können. Darunter fällt aber auch, dass das System möglichst reibungslos funktioniert und verfügbar ist. Das beinhaltet auch die aufgenommenen Daten, welche bei einem (wie auch immer gearteten) Systemausfall nicht unwiederbringlich verloren sein sollen – es sei denn, der Nutzer wünscht das so.

Da nicht nur die Systeme sicher sein müssen, sondern auch die darauf gespeicherten Daten, definiert die Kryptographie vier Hauptziele zum Schutz von Informationen (Datensicherheit, Informationssicherheit) [42]:

Vertraulichkeit: Nur berechtigte Personen dürfen Zugriff auf die gespeicherten Daten haben, Nachrichten lesen oder Informationen bezüglich ihres Inhalts erlangen. Unberechtigten Personen muss der Zugriff verwehrt bleiben – idealerweise wissen sie nicht einmal von der Existenz der Daten.

Integrität: Die Daten müssen vollständig und nachweislich unverändert sein. Änderungen sollten demnach entweder nicht möglich, oder aber nur dokumentiert erfolgen können.

Authentizität: Die Daten müssen eindeutig und überprüfbar einem Absender oder Urheber zugeordnet werden können.

Verbindlichkeit: Sind die Daten erzeugt, bzw. verschickt, sollte es dem Absender nicht möglich sein, die Urheberschaft zu bestreiten. Auch dritten gegenüber sollte die Autorenschaft nachweisbar sein.

Ein weiterer Aspekt der Sicherheit ist der *Datenschutz*. Er wiederum definiert den Schutz personenbezogener Daten und betrifft damit direkt die Persönlichkeitsrechte des Nutzenden.

Datenschutz: Der Schutz personenbezogener Daten soll Privatsphäre und Persönlichkeitsrechte, bzw. Anonymität wahren. Eine missbräuchliche Datenverarbeitung soll verhindert werden und das Recht auf informationelle Selbstbestimmung geschützt werden.

Auch im SmartAssist-Projekt (siehe 1.2.3) werden Datenschutz und Datensicherheit angesprochen [43], wobei diese Konzepte sich auch auf eine andere Systeminfrastruktur beziehen und auf das erweiterte GAL-Szenario nicht ohne Weiteres zu übertragen sind.

2.5 Anforderungsdefinition

Ein System, welches das zuvor in 2.3.3 definierte aggregierte Szenario umsetzen können soll, muss also zusätzlich die gerade in Abschnitt 2.4 erläuterten Grundsätze von Angriffssicherheit, Betriebssicherheit, Vertraulichkeit, Integrität, Authentizität, Verbindlichkeit und Datenschutz umsetzen. Während es zunächst so erscheint, als wären diese Grundsätze nur auf die tragbare Sensorik und die Basisstation anzuwenden, wird bei einer genaueren Betrachtung des Aspekts *Betriebssicherheit* und des Szenarios der *Fernüberwachung* deutlich, dass hier auch weitere Systeme mit einbezogen werden müssen.

Die Betriebssicherheit verlangt unter anderem auch eine sichere Speicherung der Daten und gewisse Vorkehrungen gegen unbeabsichtigten Datenverlust. Da im anzunehmenden Fall eines kompletten Systemausfalls, unter Umständen auch alle Daten gelöscht werden könnten, sind entsprechende Sicherungen unerlässlich. Wenn jedoch eine Sicherung am selben Ort aufbewahrt wird, ist im Falle eines Komplettverlusts (beispielsweise durch Brand oder Diebstahl) nichts gewonnen. Eine Datensicherung zu einem physikalisch entfernten Ort wäre demnach angebracht.

Ähnliches gilt auch für die Anbindung an Dienstleister, die z.B. den Betrieb eines Hausnotrufservices anbieten. Zum einen müssen die abgesetzten Notrufe ja auch angenommen werden, zum anderen müssen hier auch entsprechende Schnittstellen und Mechanismen vorhanden sein, um die bloße Funktionalität des Systems aus der Ferne überwachen zu können, um so die Verfügbarkeit des Dienstes zu überprüfen.

Auch die Fernüberwachung (beispielsweise des Reha-Sports im Szenario zur Vitalparameterüberwachung) setzt eine Anbindung an eine Infrastruktur voraus, welche in der Lage ist, die oben angeführten Anforderungen an Datenschutz, und Sicherheit zu erfüllen.

Das Ziel des umzusetzenden Systems ist es, alle Anwendungsfälle des beschriebenen aggregierten Szenarios zu unterstützen. Da das Augenmerk dieser Arbeit auf der Datenerfassung und der Datenübermittlung liegt, ist es deshalb zunächst von Interesse, was für Daten aufgezeichnet werden und welche Eigenschaften diese Daten haben. Im folgenden Kapitel soll zunächst eine Anforderungsanalyse stattfinden, die genauer beleuchtet, mit was für Sensoren und damit, mit was für Datenraten in den betrachteten Szenarien denn überhaupt zu rechnen ist.

Aus Sicht einer zuverlässigen drahtlosen Datenübertragung vom WBAN zu einer Datensenke ist es letztendlich allerdings unerheblich, von welchem Sensor die aufgezeichneten Daten stammen. Wichtig sind in jedem Fall die resultierenden Datenraten und die sich daraus ergebenden Einschränkungen und Möglichkeiten.

3 Systementwurf und mögliche Umsetzungen

> „History has taught us: never underestimate the amount of money, time, and effort someone will expend to thwart a security system. It's always better to assume the worst. Assume your adversaries are better than they are. Assume science and technology will soon be able to do things they cannot yet. Give yourself a margin for error. Give yourself more security than you need today. When the unexpected happens, you'll be glad you did."
>
> Bruce Schneier, 1997 [44]

Aus dem in Kapitel 2 definierten aggregierten Szenario (Abschnitt 2.3.3) für die zuvor erläuterten Anwendungsfälle und den sekundären Anforderungen bezüglich Sicherheit und Stabilität (Abschnitt 2.4) soll in diesem Kapitel ein vollständiges System skizziert werden.

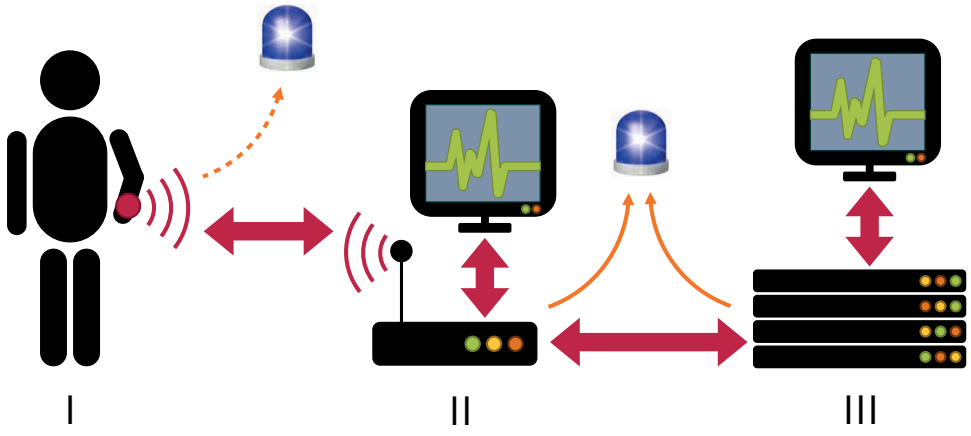


Abbildung 3.1: Gesamtsystem mit dem tragbaren Sensorsystem (I), der heimischen Basisstation (II) und den Infrastrukturkomponenten (III). Die heimische Basisstation verfügt ebenso wie die Infrastruktur über einen Notrufkanal; bei der tragbaren Sensorik ist dieser unabhängig ausgeführt.

Das in Abbildung 3.1 dargestellte Gesamtsystem besteht also aus drei physikalischen Systemen: Die am Körper eines Menschen getragenen Sensorsysteme, die Basisstation in der Wohnung des betreffenden Menschen und die Infrastruktur im Hintergrund. Außerdem ist hier auch die Datenübertragung zwischen tragbarer Sensorik und der häuslichen Basisstation sowie die Datenübertragung zwischen Basisstation und Backend-Infrastruktur dargestellt. Ein weiterer Aspekt sind die Notrufkanäle zur Alarmmeldung bei einem Notfall. Während die Basisstation und die Infrastrukturkomponenten über feste und in der Regel kabelgebundene Kanäle verfügen, ist der Notrufkanal der tragbaren Sensorik davon unabhängig zu implementieren, um auch eine Funktionalität außerhalb der Wohnung sicherzustellen.

Im Folgenden werden die beteiligten Komponenten genauer betrachtet, wobei jeweils auf vorhandene Ansätze und gegebenenfalls deren Einschränkungen eingegangen wird.

3.1 Tragbare Sensorsysteme

Ein tragbares Sensorsystem muss für die angestrebten Anwendungsfälle diverse Anforderungen erfüllen. Zum einen muss es – wie der Name schon vermuten lässt – tragbar sein, sich also ohne den Träger großartig einzuschränken am menschlichen Körper befestigen lassen. Zum anderen – und um einen gewissen Komfort zu erreichen – sollte es möglichst lange ohne Unterbrechung funktionieren; die Batterielebensdauer sollte also entsprechend lang sein. Beliebig große Batterien oder Akkumulatoren würden sich negativ auf den Tragekomfort auswirken, weswegen das System an sich stromsparend sein muss.

Im Folgenden wird kurz auf einige Exemplare aus der stetig wachsenden Familie der drahtlosen Sensorknoten eingegangen, welche sich potentiell als Basis für das tragbare Sensorsystem eignen. Außerdem werden zwei kommerzielle Produkte, die häufig in medizinischen Studien zur Aktivitätserfassung eingesetzt werden, vorgestellt.

3.1.1 Drahtlose Sensorknoten

Drahtlose Sensornetze – und damit drahtlose Sensorknoten – wurden bereits für eine Vielzahl von Anwendungen entwickelt und in diversen Forschungsprojekten eingesetzt. Vom Monitoring einer Raffinerie [45] über das Verfolgen von Wildtieren [46] bis hin zur Integritätsüberwachung von Gebäuden [47] teilen alle diese Ansätze jedoch ähnliche Einschränkungen. In der Regel kommen dabei kleine, auf Mikrocontrollern basierende Rechner zum Einsatz, die in vielerlei Hinsicht beschränkt sind. So steht meistens nur eine geringe Rechen- und Speicherkapazität zur Verfügung, die drahtlose Kommunikation ist unzuverlässig und in Durchsatz und Reichweite beschränkt und die batteriebetriebene Energieversorgung limitiert die Lebenszeit.

Der grundsätzliche Aufbau drahtloser Sensorknoten ist für alle im Folgenden betrachteten Exemplare ähnlich und in Abbildung 3.2 dargestellt: Ein *Mikrocontroller* steuert als zentrales Element alle weiteren Komponenten. Die Messwerte, die von den verschiedenen *Sensoren* aufgenommen werden, können auf (unterschiedlichen) *Speichern* abgelegt werden und/oder über einen *Funktransceiver* versendet werden. Als *Energiequelle* kommen dabei in der Regel Akkus oder Batterien zum Einsatz.

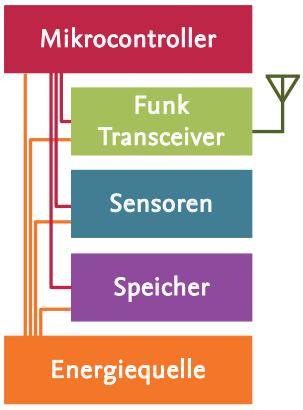


Abbildung 3.2: Bestandteile eines drahtlosen Sensorknotens.

Ein Sensorknoten muss nicht zwangsläufig über ein Betriebssystem verfügen, für eine eventuelle Portierung der entwickelten Software auf andere Sensorsysteme kann ein Betriebssystem aber sehr vorteilhaft sein, da so große Teile der Implementierung nur für ein bestimmtes Betriebssystem und nicht für eine bestimmte Hardware umgesetzt werden müssen. Für eine Portierung auf eine andere Hardware müsste dann – in der Theorie – nur die Unterstützung für das entsprechende Betriebssystem gegeben sein.

Im Folgenden werden einige exemplarische Beispiele und kommerzielle Produkte aus der Welt der Sensor-

knoten kurz vorgestellt, die ähnliche Aufgaben, wie die zuvor in Abschnitt 2.3 beschriebenen, erfüllen können. Dabei wird auch auf die jeweiligen Besonderheiten und Schwächen eingegangen.

Der TMote Sky wurde dabei als Vertreter für die MSP430-Architektur von Texas Instruments ausgewählt, welche auch die Basis für viele weitere Sensorknoten darstellt (beispielsweise Zolertia Z1, EPIC [48] oder ScatterWeb [49]). Der AVR Raven repräsentiert Atmels ATmega Architektur, welche außerdem die Basis für die FireFly [50] und MicaZ-Knoten bildet.

Der Shimmer Sensor [51] basiert ebenfalls auf der MSP430-Architektur, da er aber im Bereich der medizinischen Forschung weit verbreitet ist, wird er einer genaueren Betrachtung unterzogen.

TelosB / TMote Sky

Wenn man nach dem „Standard"-Sensorknoten sucht, wird man zwangsläufig auf den TMote Sky bzw. TelosB [52] stoßen, der von der Berkley University of California entwickelt wurde; zwischen beiden Varianten bestehen nur sehr geringe Unterschiede, weswegen an dieser Stelle nicht zwischen ihnen unterschieden werden soll. In Wissenschaft und Forschung sind diese Sensorknoten sehr weit verbreitet: Google-Scholar listet zur Zeit 3020 Publikationen, die den TMote Sky erwähnen, bzw. 3810, die den TelosB erwähnen. Außerdem sind mittlerweile zahlreiche Nachbauten erhältlich, wie der Maxfor MTM-CM500-MSP (siehe Abbildung 3.3), welches gleichzeitig die günstigste Art ist, einen solchen Sensorknoten für ∼100 Euro zu erwerben.

Die zentrale Einheit bildet hier ein MSP430-Mikrocontroller von Texas Instruments, der auf einer 16-Bit-RISC-Architektur beruht und normalerweise mit 4 MHz getaktet ist. Der Prozessor verfügt über drei verschiedene interne Speicher: 40 kByte Flash, 10 kByte SRAM und 4 kB EEPROM.

Über einen Serial Peripheral Interface (SPI)-Bus ist ein CC2420-Funktransceiver angebunden, der im 2,4-GHz-Band arbeitet; als Antenne dient dabei entweder eine Chip-Antenne, alternativ lässt sich über eine optionale SMA-Buchse auch eine externe Antenne betreiben. Derselbe SPI-Bus dient auch zum Anschluss des onBoard-Flash-Speichers. Üblicherweise sind die Knoten mit einem Luftfeuchtesensor und zwei Licht/IR-Sensoren ausgestattet, die über den Inter-Integrated Circuit (I^2C)-Bus bzw. an den prozessorinternen Analog-to-Digital Converter (ADC) angebunden sind.

Die Spannungsversorgung ist über zwei Mignonzellen (AA) realisiert, die auf der Rückseite einen Großteil der Fläche des Knotens ausmachen und die den Sensorknoten direkt mit ∼ 3 V versorgen. Über den USB-Port lässt sich der TeleosB ebenfalls mit Spannung versorgen, wobei die 5 V USB-Spannung in diesem Fall über einen DC/DC-Wandler auf 3.3 V gewandelt wird. Der USB-Port eignet sich auch zum einfachen Programmieren des Knotens – es werden also keine separaten Programmiergeräte benötigt. Als Betriebssysteme werden beispielsweise Contiki [53] und TinyOS [54] direkt unterstützt.

Im GINSENG-Projekt [45] wurde die Erfahrung gemacht, dass das dort entwickelte Time Division Multiplex (TDMA)-basierte Medium Access Control (MAC)-Protokoll GinMAC [55] nicht zeitgleich mit der IPv6-Implementierung lauffähig war, weil der Arbeits- und Programmspeicher des Prozessors schlicht nicht ausgereicht hat, um beides zu vereinen. Um eigene Übertragungsprotokolle implementieren und trotzdem noch die Grundfunktionalität aufrechterhalten zu können, ist dieser Prozessorkern demnach nicht ausreichend.

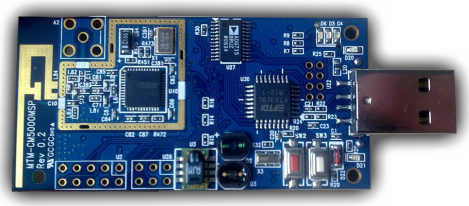


Abbildung 3.3: Der TMote Sky bzw. TelosB bzw. Maxfor MTM-CM500-MSP.

Shimmer-Sensor

Der Shimmer-Sensor [51] in Abbildung 3.4 basiert auf dem gleichen MSP430-Mikroprozessor und der gleichen Architektur wie die der gerade beschriebene TelosB-Sensorknoten, weshalb auch sein grundsätzlicher Aufbau sehr ähnlich ist. Der Sensorknoten wird häufig im Bereich der Aktivitätsüberwachung eingesetzt und ist mit einem analogen Accelerometer (Freescale MMA7260QT) ausgestattet, das an den internen Analog-Digital (AD)-Wandler des Mikrocontollers anschlossen ist, welcher die Abtastung mit einer Auflösung von 12 Bit erlaubt.

Neben dem IEEE 802.15.4-Funkinterface verfügt der Knoten auch noch über eine Bluetooth-Schnittstelle. Zur Aufzeichnung von größeren Datenmengen steht ein microSD-Kartenslot zur Verfügung, welcher in früheren Versionen des Sensorknotens nur als Alternative zum Funkinterface verwendet werden konnte. Als Betriebssystem wird TinyOS mitgeliefert.

Der Sensorknoten lässt sich für ungefähr 200 Euro erwerben; ein Daughterboard mit einem Gyroskop kostet mit ∼150 Euro ebenso viel wie ein ebenfalls erhältliches GPS-Board, welches auch einen Luftdrucksensor beinhaltet.

Im Gegensatz zum TelosB-Knoten, verfügt der Shimmer-Sensor über keinen direkten USB-Anschluss; zum Programmieren ist ein gesonderter Adapter erforderlich, der noch einmal 200 Euro kostet und der über eine proprietäre Schnittstelle angebunden ist. Über diesen Adapter lässt sich dann auch der interne Lithium-Polymer-Akkumulator (Kapazität: 280 mAh) laden.

Ein auf Shimmer basierendes System zur Erfüllung der erweiterten Anforderungen (also die Ausstattung mit Accelerometer, Gyroskop und Luftdrucksensor) würde auf der einen Seite durch zwei erforderliche Daughterboards relativ dick; auf der anderen Seite würde es mit insgesamt 700 Euro auch unverhältnismäßig teuer.

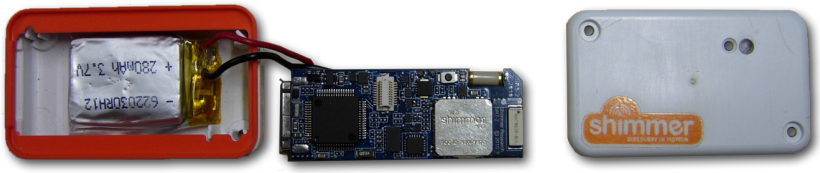


Abbildung 3.4: Die Einzelteile des Shimmer-Sensors: Gehäuseunterteil mit Akku, Basisplatine und Abdeckung.

AVR Raven

Der Atmel AVR Raven (siehe Abbildung 3.5) basiert auf Atmels ATmega1284p Mikrocontroller, der auf einer 8-Bit-RISC-Architektur beruht. Er ist normalerweise mit 8 MHz getaktet und verfügt über 128 kByte Flash, 16 kByte SRAM und 4 kByte EEPROM. Außerdem verfügt er über einen internen 10-Bit-ADC, eine Real Time Clock (RTC) und eine Vielzahl von separierten Kommunikationsbussen: Während der MSP430 bei der Verwendung unterschiedlicher Busse, den einzigen dafür zur Verfügung stehenden UART umprogrammieren muss, sind beim ATmega 1284p I^2C, JTAG, zwei SPI und UART separat ausgeführt.

Außerdem verfügt der AVR Raven mit dem ATmega 3290 über einen sekundären Mikrocontroller, der hauptsächlich für die Anbindung zahlreicher Eingabe- und Ausgabekomponenten genutzt ist: Neben LCD, Lautsprecher, Mikrofon, Joystick und Temperatursensor ist hier auch ein 16-MBit großer externer Flash-Speicher angebunden, was die Entwicklung von Anwendungen, die Daten aufnehmen und darauf speichern, verkompliziert. Die beiden Controller kommunizieren über eine serielle Verbindung (UART) miteinander.

Die Programmierung gestaltet sich dabei aufwändig, da in jedem Fall ein separates Programmiergerät vonnöten ist, um die Mikrocontroller über JTAG oder ISP zu programmieren. TinyOS und Contiki unterstützen den AVR Raven standardmäßig.

Im Gegensatz zu den meisten anderen Funktransceivern verfügt der hier zum Einsatz kommende AT86RF230 über keine Unterstützung zur AES-Verschlüsselung. Dafür ist er – wiederum im Gegensatz zu den oben genannten TMote Sky bzw. Shimmer – über einen eigenen SPI-Bus an den Mikrocontroller angebunden, was eine bessere Performance beim Empfangen und Speichern von Daten verspricht. Die Antenne ist als (relativ große) Leiterbahn (PCB) ausgeführt; eine Option für eine externe Antenne ist zwar vorgesehen, aber nicht bestückt.

Zwischen Batteriebetrieb (2 LR33-Knopfzellen) und externer Spannungsversorgung muss mit einem Jumper umgeschaltet werden: Während die Batteriespannung direkt eingespeist wird, regelt bei externer Spannungsversorgung ein DC/DC-Wandler die Versorgungsspannung auf 3.3 Volt. Die Batteriespannung wird über einen Spannungsteiler gemessen und kann so vom Sensorknoten während des Betriebs bestimmt werden.

Die Kosten sind mit ungefähr 60 Euro pro Stück relativ moderat, allerdings ist die vorhandene Sensorik für den geplanten Anwendungsfall nicht ausreichend.

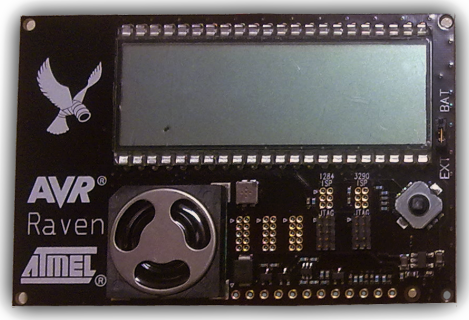


Abbildung 3.5: Der Atmel AVR Raven Sensorknoten.

iSense

Der iSense-Sensorknoten (siehe Abbildung 3.6) wird unter anderem im Wisebed-Projekt [56] und dort auch im Sensorflur [57] eingesetzt. Das Kernmodul basiert auf einem System-on-a-Chip (SoC) von Jennic, dem „JN5148 wireless microcontroller“, welches sowohl eine 32-Bit-RISC-CPU als auch einen Funktransceiver für IEEE 802.15.4 beinhaltet. Die CPU verfügt über 128 kByte an ROM- und 128 kByte an RAM-Speicher und kann zwischen 4 und 32 MHz getaktet werden. Die Antenne ist dabei entweder als PCB- oder Chip-Antenne ausgeführt; alternativ ist ein Anschluss für eine externe Antenne herausgeführt. Die Spannungsversorgung kann entweder direkt oder über einen (per Software steuerbaren) Spannungsregler erfolgen.

iSense verfolgt ein modulares Konzept: Weitere Funktionalität kann über steckbare Daughterboards hinzugefügt werden. So muss auch eine USB-Schnittstelle zunächst über ein separates Gateway-Modul zur Verfügung gestellt werden. Auch für die Spannungsversorgung gibt es – je nach Einsatzzweck – unterschiedliche Module. Über ein „Security Sensor Module“ lässt sich ein Accelerometer anbinden; das „Weather Sensor Module“ enthält neben Sensoren für Temperatur und Luftfeuchte auch einen Luftdrucksensor, dessen Auflösung allerdings eine Größenordnung zu gering ist, um ihn zum Erkennen von geringeren Höhenunterschieden einzusetzen.

Der iSense-Knoten bietet keine Unterstützung für die gängigen Betriebssysteme; als Basis-Software kommt die *iSense Firmware* zum Einsatz.

3.1.2 Kommerzielle Sensorsysteme

Im professionellen medizinischen Umfeld gibt es etliche Produkte, die für eine Aktivitätsüberwachung eingesetzt werden, wobei es sich nicht immer um „Medizinprodukte im Sinne des Medizinproduktgesetzes“

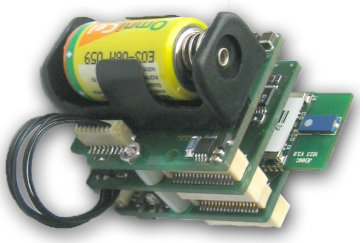


Abbildung 3.6: Der iSense-Sensorknoten mit Power- und Gateway-Modul [1].

handelt, was einer besonderen Zertifizierung bedürfen würde. Zwei häufig anzutreffende Vertreter werden im Folgenden kurz vorgestellt.

Actigraph

Der Actigraph [58] in Abbildung 3.7 ist ein weit verbreitetes Sensorsystem zur Aufzeichnung von Aktivitäten, die mittels eines Beschleunigungssensors erfasst werden. Google-Scholar listet ungefähr 12 800 wissenschaftliche Publikationen, die den Actigraph erwähnen. Die Bandbreite geht dabei von der Schlafüberwachung [59] bis zu Behandlungsansätzen bei Hyperaktivität [60].

Da über das kommerzielle Produkt kaum technische Daten öffentlich verfügbar sind, wurde ein Exemplar vollständig (und irreversibel) zerlegt, um so zumindest einige Spezifikationen zu erhalten: Der Actigraph basiert auf einem Mikrocontroller in 32-Bit-ARM-Architektur (Cortex M3 Revision 2), welcher mit bis zu 92 MHz betrieben werden kann. Der Prozessor verfügt über 256 kByte an Flash und 48 kByte an RAM. Zur Speicherung der aufgezeichneten Daten dient ein 2 GB großer NAND-Flash-Speicher. Das analoge Accelerometer ist an den internen 12-Bit-AD-Wandler des Prozessors angeschlossen, wobei die analogen Signale zuvor durch einen externen Tiefpass von 24.85 Hz geleitet werden.

Die Spannungsversorgung erfolgt über einen Lithium-Polymer-Akkumulator (Kapazität: 220 mAh), der über einen 3.3 Volt Festspannungsregler Prozessor, Speicher und Accelerometer versorgt. Der Akku kann über die USB-Schnittstelle geladen werden, die sich hinter einer schraubbaren Abdeckung befindet; das gesamte Gehäuse ist mit Dichtringen versehen und somit wasserfest.

Der Actigraph verfügt in der analysierten Version über keine Funkschnittstelle, weswegen ein Auslesen der Daten nur per USB möglich ist – und das auch nur dann, wenn ein Windows-PC mit entsprechender Software und gültiger Lizenz für diese Software vorhanden ist. Der Einzelpreis für die Hardware ist mit 255 $ relativ gering. Allerdings kostet die zum Auslesen der Daten notwendige Softwarelizenz in ihrer günstigsten Variante weitere 1295 $.

Abbildung 3.7: Der Actigraph-Aktivitätssensor [2].

[1] Quelle: `http://www.coalesenses.com`
[2] Quelle: `http://www.actigraphcorp.com`

SenseWear-Armband

Google-Scholar listet ungefähr 2280 wissenschaftliche Publikationen, die das SenseWear-Armband (Abbildung 3.8) referenzieren. So wird es zum Beispiel eingesetzt, um die sportlichen Aktivitäten von Kindern zu quantisieren [61] oder um die Alltagsaktivität von Sekretärinnen und Krankenschwestern zu überwachen [62].

Qualifizierte technische Daten des SenseWear-Armbands zu erhalten, gestaltet sich genauso schwierig wie beim zuvor erwähnten Actigraph. Da das SenseWear-Armband zu Preisen ab 1023.40 Euro zu erwerben ist, wurde von einem Kauf und einer destruktiven Analyse abgesehen.

Aus den frei verfügbaren Informationen lässt sich schließen, dass es über ein 2-Achsen-Accelerometer verfügt, welches mit einer Rate von 32 Hz abgetastet wird. Außerdem sind ein Temperatursensor und ein Hautwiderstandssensor integriert. Über eine nicht weiter spezifizierte Funktechnologie soll auch eine Funkübertragung der Daten möglich sein. Die Batterielebensdauer ist mit „bis zu 7 Tagen“ und die Speichergröße mit „bis zu 14 Tagen Aufzeichnung“ angegeben.

Abbildung 3.8: Das SenseWear-Armband [3].

3.1.3 Zusammenfassung: Vorhandene Sensorknoten und kommerzielle Systeme

Alles in allem sind die Informationen, die zu den kommerziellen medizinischen Systemen zur Verfügung gestellt werden, eher spärlich. Neben den exorbitant hohen Kosten lassen sich diese Systeme in der Regel nicht um weitere Funktionalität erweitern und es wird mit der vorhandenen Software lediglich der vom Hersteller vorgesehene Einsatzzweck unterstützt.

Letztendlich konnte auch keiner der vorhandenen Sensorknoten alle Anforderungen in Bezug auf vorhandene Sensorik, Speicherausstattung, Erweiterbarkeit, Betriebssystemunterstützung und Flexibilität erfüllen, weswegen in Kapitel 5 die Entwicklung eines eigenen Sensorknotens für den gewünschten Einsatzzweck beschrieben wird.

3.2 Datenübertragung und Datenspeicherung mit mobiler Sensorik

Wie in Kapitel 2 erläutert, soll das System in der Lage sein, sowohl Daten für eine spätere Verwendung zu speichern, als auch Daten direkt zu versenden. In diesem Anschnitt werden einige bekannte Übertragungsprotokolle und Dateisysteme erläutert und auf Ihre Eignung, diesen Anforderungen gerecht zu werden, hin analysiert.

[3] Quelle: `http://sensewear.bodymedia.com`

3.2.1 Übertragungsprotokolle

Im Bereich der drahtlosen Sensornetze gibt es eine Vielzahl von Übertragungsprotokollen. Manche sind Adaptionen aus der Welt der Personal Computers (PCs), manche wurden speziell für WSNs entwickelt und manche für spezifische Anforderungen innerhalb von WSNs. Im Folgenden werden einige dieser Vertreter kurz erläutert.

TCP/IP

Im Internet ist TCP [63] als Transportprotokoll oberhalb von **IP!** [64] das am weitesten verbreitete Protokoll. Das HyperText Transfer Protocol (HTTP) [65] setzt beispielsweise auf TCP auf und erlaubt das Aufrufen von Webseiten im World Wide Web (WWW). Wenn man „das Internet" auf Sensornetze bringen möchte, liegt es also nahe, TCP/IP für Sensornetze umzusetzen. In [66] wurde das dann auch getan, allerdings mit eher mäßigen Erfolgen: Schon in herkömmlichen drahtlosen Netzen führte die einfache Adaption von TCP zu unerwünschtem Verhalten, da TCP (in seiner ursprünglichen Variante) einen Paketverlust als Überlast interpretierte. Da aber auf dem Funkkanal ein Paketverlust mit weit größerer Wahrscheinlichkeit durch andere Faktoren hervorgerufen wird, führt diese Annahme der Überlast zu einem eher geringen Durchsatz.

Der mittlerweile aktuelle IPv6-Standard [67] bringt noch eine weitere Hürde für den Einsatz in WSNs: Durch die langen Adressen steigt der Overhead und bei relativ schmalbandigen Verbindungen kann dann allein die Adressierung schon einen Großteil der zur Verfügung stehenden Bandbreite beanspruchen (siehe dazu auch Abbildung 6.9).

6LoWPAN

Mit der Entwicklung von 6LoWpan [68] wurde diese Einschränkung dann erfolgreich angegangen. Als Akronym für „IPv6 over Low power Wireless Personal Area Networks" ist 6LoWPAN besonders für die relativ kleinen Übertragungsrahmen von IEEE 802.15.4 geschaffen worden. Es wurden Header-Kompressionsverfahren integriert und redundante Informationen reduziert. Im Idealfall kann der Overhead von UDP auf 6LoWPAN auf 7 Byte komprimiert werden. In [69] werden einige aktuelle 6LoWPAN-Umsetzungen vergleichend betrachtet.

RIME

RIME [70] wurde als Cross-Layer Kommunikationsstack für Contiki entwickelt. Ziel war es, bei geringem Overhead eine gute Performance bei der Datenübertragung in WSNs zu erreichen und dabei eine breite Funktionsvielfalt bereitzustellen. So ist RIME nicht unbedingt als *ein* Protokoll zu sehen; vielmehr können verschiedene Funktionalitäten, wie beispielsweise die Multihop-Fähigkeit, konfiguriert und genutzt werden. RIME nutzt dazu logische Kanäle, die jeweils ihre eigenen Eigenschaften und Protokollspezifikationen haben.

GINMAC

Das GINMAC-Protokoll [55] wurde im Rahmen des GINSENG-Projekts [45] entwickelt. Im betrachteten Umfeld galt es, Messdaten einer Raffinerie über ein drahtloses Sensornetzwerk zu einem Kontrollstand zu übermitteln. Diese zeitkritischen Daten mussten in einer garantierten Zeit im Netzwerk verbreitet werden. Innerhalb von 2 Sekunden sollte ein Messergebnis bei der zentralen Steuerungsinstanz eingetroffen, dort verarbeitet und eine entsprechende Aktion im Netzwerk ausgeführt sein. Erreicht wird das im konkreten Fall durch ein zeitschlitzbasiertes Protokoll, das einen festen TDMA-Baum aufspannt, und jedem Knoten explizite Zeitschlitze zuweist. Mit zusätzlichen Backup-Slots können Übertragungsfehler behandelt werden. Auf diese Weise können zwar zeitliche Bedingungen eingehalten werden, jedoch sinkt durch die statische Konfiguration der erreichbare Durchsatz rapide, da auch für Knoten, die keine Daten zu senden haben, immer Zeitschlitze reserviert sind.

Zusammenfassung: Übertragungsprotokolle für WSNs

Allen vorgestellten Protokollen ist gemein, dass sie auf eine durchgehende Ende-zu-Ende Verbindung vertrauen. Bei den Standard-Protokollen wie TCP und UDP, die aus der Welt der drahtgebundenen Kommunikation stammen, ist das auch nicht weiter verwunderlich. Bei GINMAC wird nicht nur die Ende-zu-Ende-Verbindung vorausgesetzt sondern auch eine statische Topologie.

Prinzipiell eignen sich jedoch alle vorgestellten Protokolle zum Übertragen von Daten zwischen dem WBAN und der Basisstation. Jedoch müsste für jeden Anwendungsfall eine eigene Applikation geschrieben werden, die zwischen dem Speichern und dem Übertragen von Daten entscheidet. So muss z.B. sichergestellt werden, dass Daten, die nicht versendet werden können, weil gerade kein Empfänger in Reichweite ist, entsprechend zwischengespeichert werden. Auch müssen Mechanismen zum Erkennen (*Discovery*) von möglichen Kommunikationspartnern implementiert werden, damit im Falle einer Funkverbindung, die zuvor unterbrochene Übertragung wieder aufgenommen werden kann. Des Weiteren wären für jede Anwendung Strategien zum erneutem Übertragen (im Fehlerfall) oder Verwerfen (bei doppelter Übertragung) zu entwerfen und umzusetzen.

Da keines der vorhandenen Übertragungsprotokolle den Ansprüchen an ein transparentes Handover zwischen „Speichern wenn nicht in Funkreichweite“ und „Übertragen wenn in Funkreichweite“ umsetzen kann, ohne eine entsprechende spezialisierte Anwendung dafür zu implementieren, wird in Kapitel 6 auf die Technik der unterbrechungstoleranten Netzwerke und deren Umsetzung für das Gesamtsystem eingegangen.

3.3 MSHP – die Basisstation zu Hause

Das Gateway, die Multi-Services Home Platform (MSHP), ist die zentrale Instanz innerhalb der Wohnung. Alle aufgenommen Daten werden hier gespeichert und verarbeitet. Um dem Datenschutzkonzept zu genügen, sollen alle Daten auch *nur* hier verarbeitet werden und nicht etwa bei einem Cloud-Service oder anderen Dienstleistern.

Damit nicht nur die Daten der hier beschriebenen mobilen Anwendungsfälle aufgenommen werden können, stellt die MSHP auch für alle anderen (in Abschnitt 1.1 kurz angerissenen) Anwendungsfälle Schnittstellen in Hard- und Software bereit. Dazu kommt mit der OSGi-Service-Plattform eine Middleware zum Einsatz, die für den konkreten Fall in [71] beschrieben ist.

Generell können dabei drei verschiedene Middleware-Ansätze im AAL-Umfeld unterschieden werden: Agenten-basierte Systeme wie OASIS [72], Event und Datenfluss-basierte Systeme wie PERSONA [73] und Serviceorientierte Systeme wie SOPRANO [74]. Die hier verwendete Plattform fällt damit in den letztgenannten Bereich, wobei hier im Unterschied zu SOPRANO keine Ontologie-basierte Architektur verwendet wird.

An die Hardware eines solchen Systems werden zunächst keine besonderen Anforderungen gestellt, wobei eine leise und stromsparende Umsetzung die Akzeptanz eines solchen Systems erhöhen würde. In [75] wurde die Integration der genannten Funktionalitäten in das eingebettete System eines handelsüblichen Routers für den Heimgebrauch beschrieben.

In Abschnitt 8.1 erfolgt eine genauere Beschreibung von Herausforderungen und deren Umsetzung, die dieses Gateway betreffen.

3.4 Notfallmeldungen

Um den Anforderungen an ein Hausnotrufsystem gerecht zu werden, müssen auch entsprechende Kanäle zum Absetzen von Notfallmeldungen vorgesehen werden. Im angestrebten Szenario gibt es dafür zwei verschiedene Ausgangspunkte. Zum einen muss die Basisstation in der Wohnung der jeweiligen Nutzer in der Lage sein, im Notfall auch Alarmmeldungen absetzten zu können. Alternativ kann diese Aufgabe aber auch von einem Backend-System wahrgenommen werden. Zum anderen sollte die mobile Sensorik auch in der Lage sein,

ohne Mitwirkung einer speziellen Infrastruktur Notfallmeldungen abzusetzen.

3.4.1 Alarmmeldung von der MSHP

Im GAL-Projekt wurde nicht nur auf akute Notfälle wert gelegt, welche die sofortige Reaktion eines Rettungsdienstes erfordern würden. Vielmehr wurden auch die Möglichkeiten von verschiedenen Alarmierungsstufen diskutiert und umgesetzt. So erfordert beispielsweise eine detektierte veränderte Nahrungsaufnahme oder eine erkannte Verschlechterung des Ganges nicht die sofortige Alarmierung von Rettungskräften. Die Information von Angehörigen oder Nachbarn über das veränderte Verhalten kann aber – wenn diese denn zeitnah reagieren – dazu führen, spätere Notfälle zu verhindern, indem sie etwa die Nutzung einer Gehhilfe nahelegen oder die Ursache für das veränderte Verhalten erfragen und so die Möglichkeit haben, entsprechende Maßnahmen anzustoßen, bevor die (negative) Veränderung auch negative Auswirkungen hat.

Das Weiterleiten (Routing) von Informationen oder Alarmmeldungen von der häuslichen Basisstation zu Verwandten, Rettungsdiensten oder sonstigen Einrichtungen wurde an anderer Stelle bearbeitet und veröffentlicht [76].

3.4.2 Notrufsystem für die tragbare Sensorik

Dass auch die tragbare Sensorik in der Lage sein sollte, ohne die dauerhafte Anbindung an die häusliche Basisstation einen Notruf abzusetzen und Stürze zu erkennen, wurde bereits erläutert. Entwurf, Umsetzung und Evaluation werden in Kapitel 7 beschrieben.

3.5 Backend-Systeme

Für die primäre Funktionalität erscheint die Basisstation beim Nutzer in der Wohnung in Kombination mit der mobilen Sensorik und den Alarmierungskanälen zunächst ausreichend. Wenn man aber auch Aspekte der Betriebssicherheit betrachten und so eine gewisse Dienstqualität garantieren möchte, werden zentrale Instanzen benötigt, die die verteilten und beim Endanwender zu Hause installierten Systeme überwachen und für einen dauerhaften und stabilen Betrieb sorgen.

Eine Systemüberwachung kann dabei auf verschiedenen Ebenen geschehen und es ist in jedem Fall sicherzustellen, dass durch diese die Aspekte des Datenschutzes nicht verletzt werden. Die binäre Information, dass ein Hausnotrufsystem vorhanden und prinzipiell funktionsfähig ist, ist für den Hausnotrufdienstleister allerdings essentiell.

Ein anderer Aspekt der Betriebssicherheit ist die Verfügbarkeit der Daten. Im Falle eines (wie auch immer gearteten) Systemausfalls, der mit Datenverlust einhergeht, ist eine regelmäßige Sicherung dieser Daten (Backup) unerlässlich. Wenn dieses Backup jedoch nur auf dem lokalen Rechner gespeichert ist, besteht die Gefahr, dass es bei einem Festplattenausfall ebenfalls nicht mehr verfügbar ist. Sogar ein Backup, welches auf einer anderen Festplatte oder gar einem anderen System im selben Haushalt gespeichert wird, kann durch widrige Umstände wie Diebstahl oder Brand verlustig gehen, weswegen eine Datensicherung an einem entfernten Ort sinnvoll ist.

Diese beiden Punkte werden im Kapitel 8 behandelt.

3.6 Kommunikation zwischen MSHP und Backend

Auch die Kommunikation zwischen den einzelnen Infrastukturkomponenten und der Basisstation in der Wohnung des Anwenders muss bestimmten Anforderungen genügen. Gerade wenn als Übertragungsmedium *das Internet* angenommen wird, kann dieses in keinster Weise als *zuverlässig* oder gar *sicher* angesehen werden. Im Gegenteil ist davon auszugehen, dass dieser Kanal mindestens genauso gefährdet ist, wie die Funkübertragung zwischen tragbarerer Sensorik und Basisstation. Die Aspekte einer sicheren Datenübertragung über das

Internet, welche dann auch noch den Anforderungen des Datenschutzkonzepts (siehe 1.1.6) erfüllt, werden in Kapitel 8.4 angegangen.

3.7 Konkrete Aufgaben

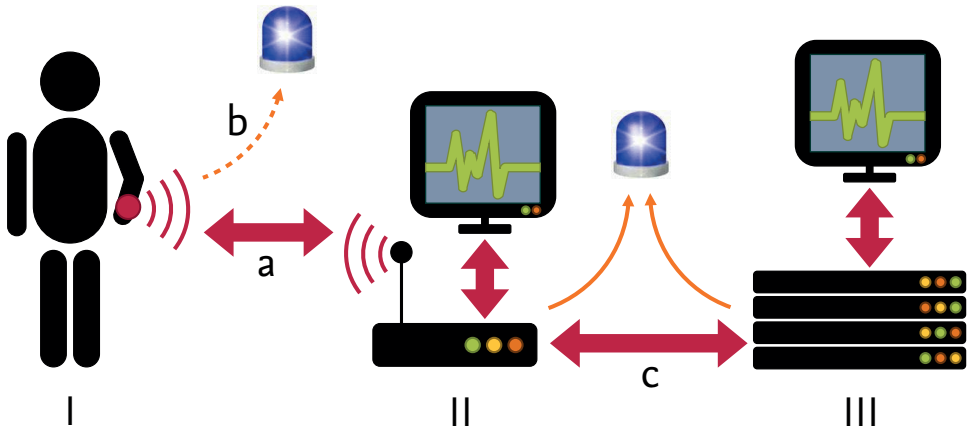


Abbildung 3.9: Gesamtsystem mit dem tragbaren Sensorsystem (I), der heimischen Basisstation (II) und den Infrastrukturkomponenten (III). Die Kommunikation zwischen Sensorsystem und Basisstation (a), und zwischen Basisstation und Infrastrukturkomponenten (c). Die heimische Basisstation verfügt ebenso wie die Infrastruktur über einen Notrufkanal; bei der tragbaren Sensorik ist dieser unabhängig ausgeführt (b).

Die konkreten Aufgaben sind mit römischen Ziffern und Kleinbuchstaben in Abbildung 3.9 markiert. Es gilt also, zunächst ein passendes Sensorsystem zu entwickeln, zu implementieren und zu evaluieren; dieses wird im Kapitel 5 ausführlich erläutert – I in der Abbildung. Eine weitere Herausforderung ist die Datenübertragung zwischen dem tragbaren Sensorsystem und dem Gateway (a in der Abbildung). Insbesondere der Fall der Synchronisation der Daten beim Verlassen des Funkradius, die Vertraulichkeit der Daten auf dem Funkkanal und eine eventuell notwendige Kompression werden im Kapitel 6 abgehandelt. Die Umsetzung des zusätzlichen mobilen Notfallkanals (b) wird zusammen mit der Möglichkeit der Sturzerkennung „auf der Straße“ in Kapitel 7 betrachtet. Die notwendigen Komponenten und Erweiterungen der Basisstationen (II) und der Backendsysteme (III) sowie die Kommunikation zwischen diesen Systemen (c) und deren Umsetzung werden in Kapitel 8 behandelt.

Zunächst gilt es jedoch in Kapitel 4, die Art und die Menge der zu übermittelnden Daten genauer zu spezifizieren.

4 Analyse: Messwerte, Sensoren und Datenraten

> „Find out why the world is not as you thought it was. Assemble the facts, digest the information, consider the implications. *Then* go spare. But with precision."
>
> Sam Vimes in *Thud!* by Terry Pratchett, 2005

Anhand der in Kapitel 3 definierten Anwendungsfälle gilt es nun, ein entsprechendes System zu entwerfen. Zunächst sind jedoch relevante Messgrößen und daraus resultierende Datenraten zu betrachten, damit das System auch praktisch nutzbar ist.

Dazu wird zunächst kurz darauf eingegangen, welche Messgrößen überhaupt aufgenommen werden. Anschließend wird anhand von Beispielen aus der Praxis dargelegt, was für Datenraten der körpernahen Sensorik in den Szenarien erwartet werden. Aufbauend auf diesen Überlegungen werden mögliche Funktechnologien auf ihre Eignung, diese Datenraten handhaben zu können, analysiert und diskutiert. Letztendlich wird eine geeignete Technologie bzw. ein geeigneter Funkstandard ausgewählt, der für den folgenden Entwurf der drahtlosen Sensorik nutzbar ist.

4.1 Der Mensch – Messwerte vom menschlichen Körper

Die Grundlage für den Entwurf von tragbaren Sensorsystemen bildet in jedem Fall der Mensch. Die Messwerte, die es aufzunehmen und zu verarbeiten gilt, basieren alle auf Parametern, die vom menschlichen Körper entweder direkt oder indirekt beeinflusst werden.

An dieser Stelle soll dabei zwischen internen und externen Parametern unterschieden werden. Als interne Parameter werden die Messwerte bezeichnet, die sich aus der Körperfunktion ergeben, also in der Regel Vitalparameter wie Blutdruck, Blutsauerstoffsättigung, Puls, EKG, Körpertemperatur, Atemfrequenz, etc.

Die externen Parameter ergeben sich in der Regel aus der Position und der Positionsänderung des Körpers; es handelt sich also um Postitions- und Bewegungsdaten. Diese können dabei sowohl durch die selbständige Bewegung des Körpers als auch durch externe Faktoren direkt beeinflusst werden.

4.1.1 Vitalparameter

Allgemein als *Vitalparameter* werden Werte bezeichnet, die zur Bewertung der Funktionen des menschlichen Körpers herangezogen werden können. In der klassischen Medizin wird dabei zwischen diskreten und kontinuierlichen Messungen unterschieden. Als *diskret* – also zu einem bestimmten Zeitpunkt abgetastet – werden dabei Blutdruck, Körpertemperatur, Atemfrequenz und Herzfrequenz bezeichnet. Im Gegensatz dazu werden EKG und Elektroenzephalografie (EEG) *kontinuierlich* aufgezeichnet. Spätestens bei Verwendung von digitaler Technik gibt es jedoch keine wirklich kontinuierliche Abtastung mehr. Vielmehr erfolgt in digitalen Systemen eine zeit- und wertdiskrete Abtastung, also immer zu diskreten Zeitpunkten und mit einer bestimmten Quantisierungstiefe. Zu unterscheiden ist insofern lediglich die Frequenz und die zur Quantifizierung verwendeten Quantisierungsstufen der aufeinanderfolgenden diskreten Abtastungen, welche dann – bei Einhaltung des Nyquist-Shannon-Abtasttheorems [77] und (sofern möglich) des Quantisierungstheorems [78] in gewissen

Grenzen – eine Rekonstruktion zu einem (bestimmten Ansprüchen genügenden) bandbegrenzten Signal erlauben.

Mit Hinblick auf die später in Abschnitt 4.2 diskutierten Datenraten werden an dieser Stelle kurz einige relevante Parameter erläutert.

Körpertemperatur

Die normale Körpertemperatur eines gesunden Menschen liegt zwischen 35.8 °C und 37.2 °C. Gemessen wird sie in der Regel mit einem Fieberthermometer, wobei der Ort der Messung (beispielsweise im Mund, unter der Achsel oder im Mastdarm) einen großen Einfluss auf Messgenauigkeit bzw. Abweichung von der realen inneren Körpertemperatur hat. Die Extremitäten weisen geringere Temperaturen auf; hier ist eine hohe Messungenauigkeit zu erwarten.

Die Umwandlung der Temperatur in eine elektrische – und damit für die elektronische Datenverarbeitung erfassbare – Größe kann auf mehrere Arten geschehen. Während *Heißleiter* bei Temperaturerhöhung einen geringeren elektrischen Widerstand aufweisen, erhöht sich der Widerstand bei *Kaltleitern* mit höherer Temperatur. Eine Spannung, die an einen solchen Heiß- oder Kaltleiter anliegt, verändert sich also mit der Temperatur und diese Änderung kann relativ einfach mit AD-Wandlern quantisiert und aufgezeichnet werden.

Herzfrequenz

Die Herzfrequenz (manchmal auch „Herzschlagfrequenz“ genannt) bezeichnet im medizinischen Kontext die Anzahl der Herzschläge pro Minute. Manuell kann man diese Frequenz durch Erfühlen des Pulses einfach selbst bestimmen. Bei einer elektronischen Messung kommen im einfachsten Fall zwei Hautelektroden zum Einsatz: Hierbei werden die sogenannten „R-Impulse“ der Herzaktivität erfasst, welches die Signale mit der höchsten Amplitude sind. Da diese Messung ohne großen Aufwand durchgeführt werden kann, basieren die im Bereich des Freizeitsports weit verbreiteten Brustgurte auf dieser Technik.

Die Herzfrequenz lässt sich aber auch optisch mit einem Pulsoxymeter bestimmen. Dieses Verfahren, das eigentlich zur Bestimmung der Sauerstoffsättigung des Blutes dient, basiert auf der Lichtabsorption beim Durchleuchten der Haut und kann so ebenfalls elektronisch ausgewertet werden.

Bei der Blutdruckmessung wird ebenfalls parallel die Herzfrequenz erfasst. Allerdings ist eine manuelle oder elektronische Blutdruckmessung nur in der Ruhelage fehlerfrei durchführbar, weswegen diese Methode bei kontinuierlichen Messungen nicht sinnvoll angewendet werden kann. Auch aus einem Elektrokardiogramm (s.u.) kann zu jeder Zeit die Herzfrequenz abgeleitet werden.

Elektrokardiogramm (EKG)

Ein Elektrokardiogramm ist ein diagnostisches Instrument und zeichnet die elektrische Aktivität der Herzmuskelfasern auf. Dabei wird der Effekt genutzt, dass jeder Muskelkontraktion des Herzmuskels eine elektrische Erregung vorausgeht. Diese Spannungsänderungen kann man mit Elektroden an der Körperoberfläche (Haut) messen, wobei im einfachsten Fall mindestens zwei Messpunkte nötig sind. Aus elektrotechnischer Sicht handelt es sich also um eine einfache Spannungsmessung. Während im stationären Einsatz meistens ein 12-Kanal-EKG verwendet wird, werden im mobilen Fall in der Regel lediglich ein bis drei Kanäle genutzt, wobei die korrekte Anordnung der Elektroden großen Einfluss auf das Messergebnis hat.

4.1.2 Postitions-, Bewegungs- und Aktivitätsdaten

Mit Sensoren zur Positions- und Bewegungserkennung lassen sich verschiedene bewegungsbasierte Aktivitäten erkennen. Für die in den Anwendungsfällen definierte Aktivitätsbestimmung, die Sturzerkennung und die Analyse von Gangparametern sind diese Daten unerlässlich. Neben den im Folgenden beschriebenen Sensoren können auch Temperatursensoren (s.o.) eine zusätzliche Auskunft über die Aktivität eines Menschen geben: So kann beispielsweise im Winter ein (vom am Körper getragenen Sensoren gemessener) Temperaturabfall

auf das Verlassen der Wohnung hinweisen.

Beschleunigungssensor

Die Beschleunigung ist die Ableitung der Geschwindigkeit; mit einem Beschleunigungssensor lassen sich also Änderungen in der Geschwindigkeit detektieren. Bei MEMS-Beschleunigungssensoren fungiert eine bewegliche Platte als Teil eines Plattenkondensators. Diese bewegliche Platte wirkt als seismische Masse, die eine Trägheit aufweist. Eine Änderung der Geschwindigkeit führt deshalb zu einer Bewegung der Platte innerhalb ihres Bezugssystems. Dies schlägt sich in einer Änderung der Kapazität des Plattenkondensators nieder, welche wiederum quantisiert werden kann.

Aktuelle Beschleunigungssensoren sind soweit integriert, dass sie eine Messung von drei (unabhängigen) Achsen gleichzeitig erlauben. So lassen sich Beschleunigungen in allen Richtungen des Raums erfassen. Die Erdbeschleunigung wirkt dabei dauerhaft auf die seismische Masse des Beschleunigungssensors, weswegen in Ruhelage auch immer ein Vektor mit der Beschleunigung von 1 g in Richtung Erdmittelpunkt bestimmt werden kann. Im weiteren Verlauf wird ein Beschleunigungssensor auch als Accelerometer bezeichnet.

Gyroskopischer Sensor

Aus Sicht eines Beschleunigungssensors findet eine Drehung um die eigene Achse nicht statt: Ein Accelerometer ist also nicht in der Lage, Drehbewegungen zu detektieren – ein Kreiselinstrument hingegen schon.

Der ursprüngliche Kreiselkompass basiert auf dem Prinzip der Drehimpulserhaltung: Wenn die Lage eines rotierenden Kreisels geändert wird, wirkt eine Kraft in die Gegenrichtung dieser Lageänderung und diese Kraft lässt sich quantifizieren.

Micromechanische Kreiselinstrumente – im weiteren Verlauf auch als „Gyroskope“ bezeichnet – basieren auf dem Prinzip des Resonator-Gyroskops. Dabei kommt allerdings kein Kreisel zum Einsatz, sondern es wird sich eine Eigenschaft der Corioliskraft zunutze gemacht: Dreht man ein vibrierendes Objekt im Raum, wird die Vibration von der Corioliskraft beeinflusst; der Einfluss steigt dabei mit der Drehrate. Im MEMS-Gyroskopen werden sogenannte „Dither Frames“ elektrostatisch in Resonanz versetzt und erzeugen so die nötige Vibration. Beim Auftreten einer Drehbewegung wird sich diese Vibration durch die Einwirkung der Corioliskraft ändern. Diese Änderung wirkt sich wiederum auf die Kapazität eines Plattenkondensators aus, der sich aus Teilen der mikromechanischen Struktur ergibt. Diese Kapazitätsänderung kann wiederum quantisiert werden.

Prinzipbedingt haben Gyroskope eine höhere Stromaufnahme als Accelerometer, da hier aktiv Resonanzen erzeugt werden müssen.

Luftdrucksensor

Mit Acclereometer und Gyroskop können bereits sechs Freiheitsgrade im Raum erfasst werden. Da ein Accelerometer nur Änderungen in der Geschwindigkeit registriert, eignet es sich nicht, um zurückgelegte Strecken, die mit gleichbleibender Geschwindigkeit zurückgelegt wurden, zu bestimmen. Der Umgebungsluftdruck kann hier eine weitere Unterstützung zur Erkennung von Position und Bewegung im Raum liefern: Mit zunehmender Entfernung vom Meeresspiegel (nach oben) sinkt der Luftdruck. Aktuelle Luftdrucksensoren sind dabei so präzise, dass bereits Änderungen von 0.1 Pa detektiert werden können, was eine Höhenänderung von wenigen Zentimetern entspricht. Da auch die Wetterlage einen Einfluss auf den vorherrschenden Luftdruck hat, lassen sich (ohne aufwändige Kalibrierung) keine Aussagen über die absolute Höhe treffen; über eine relative Höhenänderung allerdings schon. So kann beispielsweise erkannt werden, ob ein Aufzug nach oben oder unten fährt, oder ob eine Treppe hinauf oder hinab gegangen wird.

Die Funktionsweise von mikromechanischen Luftdrucksensoren ist vergleichsweise einfach: Eine Silizium-Membran weist einen spezifischen elektrischen Widerstand auf. Bei Verformung dieser Membran durch die Einwirkung von Druck, verändert sich auch der elektrische Widerstand, was wiederum durch eine

entsprechende Beschaltung (Messbrücke) detektiert werden kann.

4.2 Körpernahe Sensorik – Art und Datenraten der Sensoren

Bei der Überwachung von Vitalparametern oder sonstigen am Körper aufgenommenen Daten wird zunächst von einer Abtastung mit bestimmter Frequenz und mit bestimmten Quantisierungsstufen ausgegangen. Je nach aufzunehmender Messgröße liefern so unterschiedliche Sensoren auch unterschiedliche Datenraten. An dieser Stelle sollen lediglich einige allgemeine Beispiele und im Speziellen – etwas vorausgreifend – die Sensorik des Sensorknotens INGA (siehe Kapitel 5) betrachtet werden. Bei den betrachteten Datenraten ist generell anzumerken, dass viele Sensoren keine ganzen 2er Potenzen als Auflösung anbieten. So liefern einige Sensoren z.B. eine Auflösung von 10 Bit, also 1024 diskrete Quantisierungsstufen, welche intern dann in 2 Byte (16 Bit) repräsentiert werden. Bei der Berechnung der Nutzdatenraten gehen wir davon aus, dass auch nur die tatsächlich erzielte Auflösung übermittelt werden muss und vom Benutzer entsprechend kodiert wird.

4.2.1 Accelerometer – ADXL 345

In einer ausführlichen Analyse [79] (siehe auch Kapitel 5.2) ging der Beschleunigungssensor von Analog Devices in Bezug auf Signalrauschen und Energieverbrauch als Favorit hervor, weswegen er bei der Konzeption von INGA zum Einsatz kam. Als digitaler Sensor erlaubt er die Abtastung von 3 Achsen in zahlreichen diskreten Abtastraten im Bereich von 0.1 bis 3200 Hz, bei einer Auflösung von 10 bis 13 Bit. In der Konfiguration mit der geringsten Datenrate würde dieser Sensor eine Nutzdatenrate von $0.1\,\text{Hz} \cdot 3\,\text{Achsen} \cdot 10\,\text{Bit} = 3\,\text{Bit/s}$ erzeugen. Die höchstmögliche Datenrate läge demnach bei $3200\,\text{Hz} \cdot 3\,\text{Achsen} \cdot 13\,\text{Bit} = 124\,800\,\text{Bit/s}$. Bei Fokussierung auf den medizinischen Bereich sind beide Extrema allerdings unbrauchbar. In [80] wurde argumentiert, dass am menschlichen Körper keine Beschleunigungen $> 20\,\text{Hz}$ auftreten. Unter Einhaltung des Nyquist-Shannon-Abtasttheorems und mit etwas Sicherheitsabstand wurde in [79] 50 Hz als eine sinnvolle Abtastrate für Beschleunigungssensoren im medizinischen Bereich proklamiert. Mit dieser Abtastrate würde dann eine typische und realistische Datenrate von 1500 Bit/s erreicht.

4.2.2 Gyroskop ST L3G4200D

Das 3-Achsen Gyroskop erlaubt die Detektion von drei weiteren Freiheitsgraden bei einer Genauigkeit von bis zu 2000 Grad pro Sekunde. Mit einer Auflösung von 16 Bit und einer diskret auf 100, 200, 400 oder 800 Hz einstellbaren Abtastrate, liefert das Gyroskop Datenraten im Bereich von 4800 bis 38 400 Bit/s. Im betrachteten medizinischen Umfeld wurden bisher erst wenige Versuche mit Gyroskopen gemacht. Während die Autoren von [81] eine Abtastrate von 200 Hz verwenden, wird in [32], [82] und [31] jeweils mit 100 Hz abgetastet. Da die Autoren von [83] zeigen, dass bei einer Samplingfrequenz von 200 Hz auch eine Kalibrierung während der Benutzung möglich ist, soll diese Frequenz für das Gyroskop als typischer Wert dienen. Die daraus resultierende typische Datenrate beträgt demnach 9600 Bit/s.

4.2.3 Luftdrucksensor BMP085 (inkl. Temperatursensor)

In [37] wurde gezeigt, dass sich ein Luftdrucksensor zur Höhenbestimmung innerhalb von Gebäuden und somit zur Aktivitätserkennung eignet. In [84] wird ein solcher Sensor auch zur Sturzerkennung eingesetzt und dabei mit 1.8 Hz bei einer Auflösung von 19 Bit abgetastet, was einer zu Datenrate von 34.2 Bit/s führt. Der auf INGA verwendete Bosch-Sensor braucht zum Sampeln in höchster Auflösung (19 Bit) 25.5 Millisekunden Zeit, was zu einer maximalen Abtastrate von 39.2 Hz führt und dabei bis zu 741 Bit/s an Daten produzieren würde. Der integrierte Temperatursensor hat eine Auflösung von 16 Bit und liefert nach 4.5 Millisekunden einen Messwert, was zu einer maximalen Abtastrate von 222 Hz führt und dabei 3552 Bit/s an Daten verursachen würde. Da sich sowohl Temperatur als auch Luftdruck nicht so schnell ändern, als dass es Sinn ergeben würde, sie mit solchen Frequenzen abzutasten, wird eine Abtastung mit 2 Hz und den daraus resultierenden Daten

von 2 Hz · 19 Bit + 2 Hz · 16 Bit = 70 Bit/s als sinnvoll erachtet.

4.2.4 Puls und EKG

Nicht auf INGA integriert, aber im medizinischen Bereich häufig anzutreffen, sind die Messung von Puls und Elektrokardiogramm. Die höchste jemals beobachtete Herzfrequenz ist 1511 Schläge pro Minute (∼ 25 Hz) – bei einer Etruskerspitzmaus [85]. Bei Menschen werden Herzfrequenzen mit mehr als 220 Schlägen pro Minute nicht erwartet und somit sollte eine Auflösung von 8 Bit völlig ausreichend sein. Da sich die Herzfrequenz nicht innerhalb eines Herzschlags ändert, wäre eine Übertragung öfter als ein Mal pro Sekunde (1 Hz) auch nicht vernünftig.

Beim EKG fallen dagegen schon deutlich größere Datenmengen an. In [86] wird ein drahtloses und tragbares 1-Kanal-EKG beschrieben, das bei 8 Bit Auflösung mit einer Samplingrate von 500 Hz abtastet und so eine Datenrate von 4000 Bit/s generiert. Das mobile 3-Kanal-EKG in [87] tastet jeden Kanal mit einer Rate von 125 Hz und einer Auflösung von 14 Bit ab, was zu einer Gesamtdatenrate von 1750 Bit/s führt. Sollen hingegen die Rohdaten eines kompletten 12-Kanal EKGs mit einer Abtastrate von 500 Hz und einer Quantisierungsauflösung von 16 Bit übertragen werden, wäre eine Datenrate von 96 kbit/s die Folge. Ein typisches im medizinischen Umfeld eingesetztes EKG verfügt über eine Bandbreite von bis zu 100 Hz, was zu einer Abtastrate von 200 Hz führen würde; bei drei Kanälen würde sich hierbei eine Gesamtdatenrate von 8400 Bit/s ergeben.

4.2.5 Zusammenfassung: Erwartbare Datenraten

Tabelle 4.1: Minimale, maximale und typische Datenraten exemplarischer Sensoren.

Sensor	D_{min}	D_{max}	D_{typ}
Accelerometer	3 Bit/s	124 800 Bit/s	1500 Bit/s
Gyroskop	4800 Bit/s	38 400 Bit/s	9600 Bit/s
Luftdrucksensor	35 Bit/s	4392 Bit/s	70 Bit/s
Puls	8 Bit/s	32 Bit/s	16 Bit/s
EKG	800 Bit/s	96 000 Bit/s	8400 Bit/s

Wie aus Tabelle 4.1 zu erkennen ist, sind die Datenraten von Sensor zu Sensor und von Anwendungsfall zu Anwendungsfall verschieden. So würde beispielsweise die geringste Datenrate beim alleinigen Übermitteln von Puls-Informationen mit der Frequenz von 1 Hz bei gerade einmal 8 Bit/s liegen, während ein 3-Achsen-Accelerometer auch mehr als 124 kBit/s erzeugen kann. Durch Kombination mehrerer Sensoren lassen sich sowohl Datenraten zwischen diesen beiden Extrema als auch darüber erzeugen.

Es kann davon ausgegangen werden, dass die verwendete Funktechnologie und die Leistungsfähigkeit der tragbaren Sensorik die Datenrate begrenzen, da die Basisstation quasi beliebig dimensioniert werden kann und so bei der Datenverarbeitung keinen Flaschenhals darstellt.

4.3 Datenraten auf dem Funkkanal

Um die gerade diskutierten Datenraten übertragen zu können, bieten sich prinzipiell mehrere Technologien an, welche sich in Komplexität, Reichweite, Energiebedarf und Datenrate unterscheiden. Um die Daten von körpernahen Sensoren eines BAN zu übertragen ist eine Funkverbindung unumgänglich; kabelgebundene Lösungen würden zu erheblichen Komforteinbußen und zu signifikant erhöhten Unfallrisiken führen.

Prinzipiell wäre auch eine Nutzung von Mobilfunknetzen möglich, wenn jedoch kontinuierlich Daten übermittelt werden sollen, würde das gleich zu mehreren Einschränkungen führen:

Netzabdeckung: Gerade innerhalb von Wohnungen ist der Mobilfunkempfang (besonders bei den E-Netzen im GSM-1800-Band mit hoher Freiraumdämpfung und geringerer Transmission) teilweise mangelhaft.

Kontrolle: Ein wohlmöglich sicherheitsrelevantes System würde durch zusätzliche Akteure (Netzbetreiber, Mobilfunkbetreiber etc.) anfälliger gegen Störungen (wie z.B. Netzausfall, Dienstausfall etc.).

Kosten: Mit heutigen volumenbasierten Datentarifen lassen sich die Kosten zwar in Grenzen halten, aber unerheblich werden sie deshalb noch nicht.

Energieverbrauch: Technologiebedingt wird beim Senden die meiste Energie benötigt. Kontinuierliches Senden über ein Mobilfunknetz würde sich sehr negativ auf die Lebensdauer der mobilen Energiequelle auswirken.

Komplexität, Sicherheit und Datenschutz: Daten von einer Person in einer Wohnung über ein Mobilfunknetz, ein Backbone-Netz und wieder zurück zu einer Basisstation in der Wohnung zu senden, erhöht die Komplexität, die Fehleranfälligkeit und die Kompromittierbarkeit des Gesamtsystems.

Es gilt also, eine lokale Funktechnologie auszuwählen, die die folgenden Eigenschaften erfüllt:

- Kosten: Die Technologie sollte möglichst frei von Lizenzkosten oder Nutzungsgebühren sein.
- Datenrate: Es sollte eine ausreichende Datenrate für die Anwendungsfälle unterstützt werden.
- Energieeffizienz: Um eine lange Lebensdauer der tragbaren Sensorik zu erzielen, sollte die Funktechnologie möglichst wenig Strom aufnehmen.

4.3.1 Frequenzbänder

Da die Funkverbindung sowohl im häuslichen als auch im medizinischen Umfeld zum Einsatz kommen soll, drängt sich die Nutzung eines Frequenzbereichs in einem sogenannten ISM-Band („Industrial, Scientific, and Medical“) auf. Speziell der Frequenzbereich nach FreqBZPV, Teil B: NB D150 (siehe Tabelle 4.2), sollte hierbei Anwendung finden, da er explizit von der Bundesnetzagentur „für industrielle, wissenschaftliche, medizinische, häusliche oder ähnliche Anwendungen“ [88] vorgesehen ist.

Tabelle 4.2: Frequenzteilbereiche nach FreqBZPV, Teil B: NB D150 (aus [88]).

a.)	9 kHz	bis	10 kHz
b.)	13 553 kHz	bis	13 567 kHz
c.)	26 957 kHz	bis	27 283 kHz
d.)	40.66 MHz	bis	40.70 MHz
e.)	433.05 MHz	bis	434.79 MHz
f.)	2400 MHz	bis	2500 MHz
g.)	5725 MHz	bis	5875 MHz
h.)	24.00 GHz	bis	24.25 GHz

Prinzipiell ist jeder dieser Frequenzbereiche für eine Datenübertragung nutzbar, jedoch ergeben sich die ersten Einschränkungen schon aus der Physik. So sagt schon das Shannon-Hartley-Gesetz [89] aus, dass die geringen Bandbreiten der in Tabelle 4.2 aufgeführten Frequenzbänder a und b selbst unter idealen Bedingungen nicht in der Lage sind, mehrere 10 kBit/s an Daten zu übertragen. Bei sehr hohen Frequenzen ist wiederum eine Sichtverbindung nötig, um überhaupt Daten übertragen zu können.

Bei allen Frequenzbereichen gibt es zudem auch Einschränkungen, was die Auslastung des Frequenzbandes angeht. So ist es in der Regel nicht zulässig, die komplette Bandbreite dauerhaft zu belegen.

Während es in allen ISM-Bändern viele proprietäre Ansätze zur Übertragung von Daten gibt, haben sich im 2.4 GHz-Band bereits eine Vielzahl an Standards angesiedelt; auf einige davon wird im nächsten Teilabschnitt kurz eingegangen.

4.3.2 Übertragungsprotokolle und Standards

Eine Vielzahl von bekannten Funkstandards und Protokollen nutzen die ISM-Bänder. Gängige Standards sind zum Beispiel IEEE 802.11 (WLAN), IEEE 802.15.1 (Bluetooth) und IEEE 802.15.4 (z.B. ZigBee).

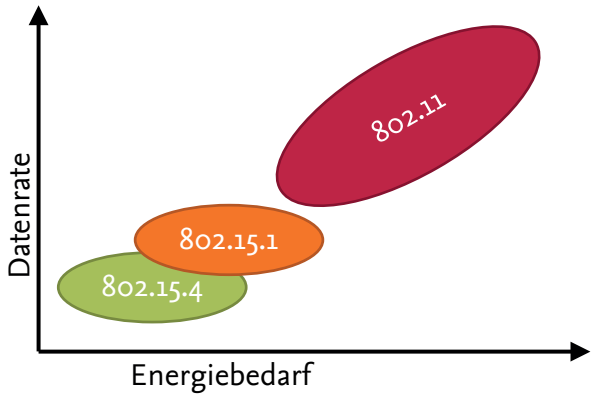


Abbildung 4.1: Energiebedarf und Datenrate ausgewählter Funktechnologien.

WLAN

Was heute gemeinhin als WLAN oder WiFi bezeichnet wird, hat seinen Ursprung im IEEE 802.11-Standard [90] von 1997, der eine Brutto-Datenübertragungsrate von 1 oder 2 MBit/s aufwies. Neben der Datenübertragung im 2.4 GHz-Band war auch noch eine Übertragung im Infrarotbereich spezifiziert. Mit der Zeit wurden etliche Verbesserungen standardisiert, welche unter anderem auch höhere Datenraten (ab IEEE 802.11b bis zu 11 MBit/s bis hin zu IEEE 802.11ac mit bis zu 1.3 GBit/s) und andere Frequenzbereiche (in IEEE 802.11a das 5 GHz-Band, in IEEE 802.11ad das 60 GHz-Band) mit sich brachten. Die Funkreichweiten unterscheiden sich je nach Frequenzband; es kann aber von Reichweiten zwischen 10 und 100 m ausgegangen werden – mit Richtantennen können im Außenbereich auch mehrere Kilometer überbrückt werden.

Allen IEEE 802.11-(WLAN-)Standards und Weiterentwicklungen (a/b/g/n/...) ist eine relativ hohe Datenrate gemein, die aber auch mit einem relativ hohen Energieverbrauch einhergeht, weswegen sie für ein WBAN nicht geeignet sind. Da Datenraten im Bereich von mehreren Megabit pro Sekunde bei köpernaher Sensorik jedoch nicht erwartet werden, wird an dieser Stelle von einer genaueren Betrachtung abgesehen.

Bluetooth

Anfänglich war Bluetooth (IEEE 802.15.1 [91]) für Bruttodatenraten bis zu 732.2 kBit/s spezifiziert. Mit der Version *Bluetooth 1.2* gab es eine geringfügige Erhöhung auf 1 MBit/s, was mit *Bluetooth 2.0 + EDR* wiederum auf 2.1 MBit/s erhöht wurde. Es sind drei Leistungsklassen definiert, die die Stromaufnahme und die Sendereichweite bestimmen. Klasse 1 sendet mit bis zu 100 mW bis zu 100 Meter weit, Klasse 2 mit 2.5 mW zwischen 10 und 50 Meter und Klasse 3 mit 1 mW zwischen 1 und 10 Meter.

In seinen früheren Versionen wies Bluetooth denselben Nachteil des hohen Energiebedarfs wie bei WLAN auf. Mit der aktuellen Version (Bluetooth 4.0) wurde dieses Manko durch eine extrem verkürzte Verbindungsaufbauzeit jedoch erfolgreich beseitigt und es ist mittlerweile möglich, relativ hohe Datenraten (brutto bis 1 MBit/s, netto bis 270 kBit/s) relativ batterieschonend zu übertragen – allerdings bleiben andere Einschränkungen: Ein Bluetooth-Piconetz ist auf einen Master und sieben Clients beschränkt, was für viele heutige und zukünftige WBANs einfach nicht ausreicht. Außerdem sind keine (wirklichen) vermaschten Netze (Mesh-Networks) möglich und eine Multihop-Kommunikation ist auch nicht vorgesehen.

Für einzelne Geräte kommt Bluetooth in der Medizintechnik inzwischen des Öfteren zum Einsatz – meist als transparente Funkschnittstelle, um früher kabelgebundene Geräte nun drahtlos anbinden zu können [92].

IEEE 802.15.4

Der IEEE 802.15.4-Standard [93] beschreibt ein Übertragungsprotokoll für Wireless Personal Area Networks (WPAN), wobei lediglich die beiden unteren Schichten des OSI-Modells definiert werden und von der Sicherungsschicht auch nur der Medienzugriff (MAC-Sublayer) implementiert ist. Die darüberliegenden Protokollebenen (Vermittlungsschicht und Transportschicht, zuständig für Wegewahl und Ende-zu-Ende-Kommunikation, sowie die Anwendungsschicht) werden separat definiert. Das macht den Standard interessant für zukünftige Entwicklungen, da hier oberhalb des MAC-Sublayers dem Entwurf und der Implementierung keine Grenzen gesetzt sind. So existieren bereits viele verschiedene Kommunikationsstacks für unterschiedliche Aufgaben: Beispiele hierfür sind ZigBee [94], 6LoWPAN [68] und der RIME-Stack [70], wobei ZigBee ein kommerzielles Ziel verfolgt und ein Lizenzmodell aufweist, welches Kosten und Einschränkungen mit sich bringt.

Die nominelle Datenrate von IEEE 802.15.4 ist mit 250 kBit/s im 2.4 GHz-Band nicht gerade üppig bemessen, jedoch liegt der Fokus von IEEE 802.15.4 vor allem auf energiesparender Umsetzung, was für batteriebetriebene tragbare Systeme von besonderer Wichtigkeit ist. Es gibt 16 überlappungsfreie Kanäle, zusätzliche Unterscheidungen können über eine PAN-ID getroffen werden, was eine große Anzahl an Netzen auf kleinem Raum ermöglicht. Neben der Übertragung im 2.4 GHz-Band sind auch noch die Frequenzbänder von 868 bis 868.6 MHz für Europa bzw. von 902 bis 928 MHz für die USA spezifiziert, wobei hier die Bruttodatenraten mit 20 bzw. 40 kBit/s für die anvisierten Anwendungsfälle nicht ausreichend sind.

Die Nettodatenrate ist jedoch abhängig von der Implementierung der darüberliegenden Protokollebenen, so können bei ZigBee unter idealen Bedingungen bis zu 110 kBit/s erreicht werden [95]. Ähnliches wurde in [96] in einem Multihop-Szenario über bis zu 6 Hops erreicht, wobei hier reine MAC-Frames versendet wurden; ein einfacher Hop konnte hier jedoch mit knapp 220 kBit/s überwunden werden, was sehr nahe am theoretischen Optimum liegt.

Tabelle 4.3: Eigenschaften der unterschiedlichen Funkstandards und Technologien.

	IEEE 802.15.4	IEEE 802.15.1 Bluetooth	IEEE 802.11 WLAN
Frequenzbäder	2.4 GHz, 868 MHz, 915 MHz	2.4 GHz	2.4 GHz, 5 GHz, 60 GHz
Gleichzeitige Netze[1]	16	∼10	3[2]
Bruttodatenrate	250 kBit/s	bis zu 2.1 MBit/s	bis zu 1.3 GBit/s
Topologie	beliebig	Stern	Stern, Ad-hoc P2P
Anzahl der Knoten	65536	8	∼32
Multihop	ja	nein	nein
Stromaufnahme	sehr gering	gering	hoch
Reichweite	10 bis 300 m	1 bis 100 m	10 bis 100 m

In Tabelle 4.3 sind die gerade erläuterten Technologien noch einmal gegenübergestellt. Der IEEE 802.15.4-Standard im 2.4-GHz-Band scheint wie geschaffen für die Datenübertragung auf der Funkstrecke zwischen der tragbaren Sensorik und der häuslichen Basisstation. Die geringe Stromaufnahme und die Flexibilität der

[1] Bei 2.4 GHz.
[2] 16 Kanäle, aber nur 3 überlappungsfrei umsetzbar.

Anwendung sind dabei die am schwersten wiegenden Argumente. Die Datenrate ist nicht exorbitant hoch, jedoch zumindest ausreichend für jeden der in Abschnitt 4.2 vorgestellten Sensoren - wenn auch (bei höchster Abtastrate) nicht für alle gleichzeitig. Um Übertragungsbandbreite (und damit auch Energie) zu sparen, kann, wenn nötig, auch noch auf eine Datenreduktion (siehe auch Abschnitt 6.3) zurückgegriffen werden.

5 Tragbare Sensorik

„The chief function of the body is to carry the brain around."

Thomas Alva Edison

In diesem Kapitel wird die Entwicklung, die Umsetzung und die Evaluation der tragbaren Sensorsysteme beschrieben. Nachdem in Abschnitt 3.1.1 schon auf den allgemeinen Aufbau von Sensorknoten und die konkreten Eigenschaften einiger bekannter Sensorknoten eingegangen wurde, soll in diesem Kapitel der speziell für Arbeiten im anvisierten Umfeld entwickelte Sensorknoten *Inexpensive Node for General Applications (INGA)* genauer beleuchtet werden, wobei die konkreten Ausstattungsmerkmale von den zuvor analysierten Sensorknoten inspiriert sind.

Da INGA nicht nur im beschriebenen Szenario eingesetzt werden soll, sondern auch noch universelleren Kriterien, wie dem Einsatz in der Lehre und der Verwendung in anderen Forschungsvorhaben, genügen muss, werden an dieser Stelle vorab einige allgemeine Kriterien aufgelistet, die der Sensorknoten zu erfüllen hat:

State of the Art: Es sollen verfügbare und bekannte Bauteile eingesetzt werden, für die es bereits eine Unterstützung gibt; auf veraltete Komponenten sollte dagegen verzichtet werden.

Erweiterbarkeit: Es sollen möglichst alle (ungenutzten) Ports und Busse für spätere Erweiterungen zur Verfügung stehen.

Handhabbarkeit: Standardlösungen sollten bevorzugt werden; es sollten möglichst keine proprietären Verbinder verwendet werden.

Einfachheit: Die Basisfunktionalität sollte niemanden abschrecken oder überfordern.

Kompatibilität: Vorhandene Betriebssysteme sollten lauffähig sein.

Kosteneffizienz: Um eine breite Anwendung zu finden, sollte der Sensorknoten so günstig wie möglich sein.

Open Source: Quelloffene Hard- und Software führt in der Wissenschaft zu größerer Akzeptanz und damit zu einer verbreiterten Nutzung.

Dazu wird zunächst die Auswahl der entsprechenden Sensoren diskutiert. Im weiteren Verlauf des Kapitels werden auch die Anforderungen an die weiteren Komponenten herausgearbeitet und so die konkreten Spezifikationen für INGA beschrieben. Nachdem ein Überblick über die Architektur und Funktionsweise gegeben wurde, wird INGA evaluiert.

5.1 Terminologie: Messaufnehmer und Sensoren

In der Mess- und Elektrotechnik ist ein *Sensor* synonym zu einem *Messaufnehmer* zu sehen, er wandelt also eine physikalische Größe in eine elektrisch messbare Größe um. In der Medizin werden *Sensoren* oft als komplette *Hardware-Software-Systeme* verstanden, die bestimmte Messgrößen aufnehmen, verarbeiten und ausgeben. In der Mikrocontroller-Praxis neigt man dazu, das entsprechende *Bauteil* als *Sensor* zu bezeichnen. Diese letztgenannte Terminologie soll auch hier verwendet werden.

Folglich gibt es auch *digitale* und *analoge Sensoren*, wobei beide Bauteile bereits Verstärkerschaltungen, Bereichsanpassungen, Filter etc. integriert haben können. Ein analoger Sensor liefert analoge Spannungen als Ausgangssignal, die anschließend abgetastet und quantifiziert werden müssen, um sie weiterzuverarbeiten. Ein digitaler Sensor hat den Analog-Digital-Umsetzer integriert und liefert die abgetasteten Messwerte über einen digitalen Datenbus; in der Regel über einen I^2C- oder SPI-Bus.

5.2 Beschleunigungssensoren

Beschleunigungssensoren wurden und werden in der Literatur am häufigsten genutzt, um Sturzerkennungen, Ganganalysen und Aktivitätserkennungen mit am Körper getragener Sensorik durchzuführen. Dabei steht in vielen Veröffentlichungen zwar, welcher Sensortyp von welchem Hersteller für die Messungen, Studien oder Versuche benutzt wurde, aber nie, warum es gerade dieser oder jener war. Wenn man die Autoren danach fragt, lautet die schlichte Antwort oft: „Weil er auf der verwendeten Plattform verfügbar war“, oder „Darüber habe ich mir keine Gedanken gemacht.“ Tatsächlich gibt es eine Vielzahl an verschiedenen Sensoren und in der früheren Veröffentlichung [79] haben wir intensive Untersuchungen zur Auswahl solcher Sensoren durchgeführt.

Beim Entwurf eines eigenen Sensorknotens sollte also zunächst die Auswahl des am besten geeignetsten Beschleunigungssensors stehen. Da außerdem viele Messungen auf dem mittlerweile nicht mehr verfügbaren *MMA7260QT* von *Freescale* basieren, war ein weiteres Ziel dieser Untersuchung herauszufinden, inwieweit die vorhandenen Ergebnisse auf andere Sensoren übertragbar sind, also wie hoch die Korrelation der getesteten Sensoren ist.

5.2.1 Sensorauswahl

Es wurden insgesamt sechs verschiedene Sensoren von drei bekannten Herstellern getestet, wobei zur Evaluation jeweils fünf Exemplare desselben Typs herangezogen wurden. In Tabelle 5.1 sind die grundsätzlichen Eigenschaften gegenübergestellt.

Tabelle 5.1: Überblick die evaluierten Accelerometer – Daten aus den jeweiligen Datenblättern.

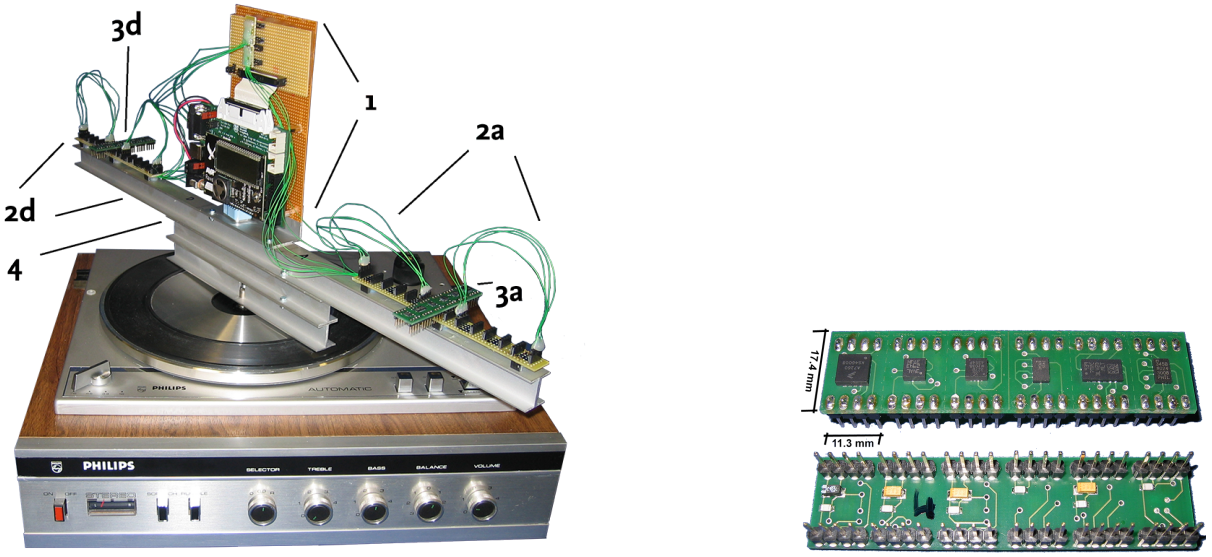
Typ		MMA7260QT Freescale	ADXL327 Analog Devices	LIS344AL ST Microelectronics	MMA7455L Freescale	ADXL345 Analog Devices	LIS3LV02DL ST Microelectronics
Auslegung		analog[1]			digital		
Abtast-	±1.5 g	x	–	–	–	–	–
auflösung(Bit)	±2 g	x	x	–	8	10	12
bei wählbaren	±3.5 g	–	–	x	–	–	–
Wertebereichen	±4 g	x	–	–	8	11	–
	±6 g	x	–	–	–	–	12
	±8 g	–	–	–	8, 10	12	–
	±16 g	–	–	–	–	13	–
Abtastrate (kHz)		≤ 11	≤ 1.6	≤ 2	≤ 0.25	≤ 3.2	≤ 2.56
Spannungsbereich (V)		2.2-3.6	1.8-3.6	2.7-3.3	2.4-3.6	2.0-3.6	2.2-3.6
Typ. Stromaufnahme (μA)		500	350	690	400	130	650
Bus (I^2C/SPI)		–	–	–	x/x	x/x	x/x

[1] Abtastrate und Auflösung der analogen Sensoren sind vom verwendeten ADC abhängig.

5.2.2 Testumgebung

In der Literatur variiert die Abtastfrequenz, mit der Accelerometer im anvisierten Anwendungsbereich abgetastet werden, sehr stark. Während in [29] mit 30 Hz abgetastet wurde, tasten die Autoren von [22] mit 1 kHz ab, um die abgetasteten Werte anschließend durch einen 250 Hz-Tiefpass zu filtern. Bei vielen weiteren Veröffentlichungen wird auf die Angabe dieser Parameter komplett verzichtet. Die Abtastrate ist nach oben nur durch die Leistungsfähigkeit des verwendeten (internen oder externen) Analog-Digital-Wandlers (ADC) beschränkt, wobei eine sehr hohe Abtastrate zu einem sehr hohen Datenaufkommen führt, was in mobilen und drahtlosen Szenarien schnell zu Problemen wie unzureichendem Speicher oder Überlastung auf dem Funkkanal führen kann. In Kapitel 4 wurde 50 Hz als sinnvolle Abtastrate postuliert, welche deshalb auch für den beschriebenen Test zum Einsatz kommt.

Um die für den anvisierten Einsatzort am menschlichen Körper bestmögliche Sensitivität zu erhalten, wurde $\pm 2g$ bei allen Sensoren als Wertebereich festgelegt, bzw. beim LIS344AL $\pm 3,5g$, da das der kleinstmögliche Wertebereich ist. Die Auflösung der Messewerte hängt von den jeweils verwendetet ADCs ab. Bei den analogen Sensoren haben wir dazu auf den internen ADC des Atmel AVR Raven Mikrocontrollers zurückgegriffen (ATmega 1284p), der eine Auflösung von 10 Bit bietet. Bei den digitalen Sensoren wurde die jeweils höchstmögliche Auflösung gewählt, das heißt 8 Bit beim MMA7455L, 10 Bit beim ADXL345 und 12 Bit beim LIS3LV02DL.



(a) Plattenspieler mit Evaluationsaufbau

(b) Versuchsträger mit 6 verschiedenen Accelerometern

Abbildung 5.1: Versuchsaufbau zur Evaluation der Accelerometer. Der primäre Teil des Evaluationsboards (1), jeweils ein analoger (2a) und ein digitaler (2d) sekundärer Teil des Evaluationsboards auf einem 30 cm Ausleger (4). Die Versuchsträger (3a und d) sind rechts vergrößert dargestellt.

Um die Sensoren alle denselben Beschleunigungen aussetzen zu können, wurde ein Plattenspieler zweckentfremdet. Abbildung 5.1 zeigt den Versuchsaufbau, der aus mehreren Teilen besteht. An zentraler Stelle ist ein Atmel AVR-Raven Sensorknoten verbaut, der die jeweils zu testenden Sensoren abfragt. Während die digitalen Sensoren per SPI-Bus angeschlossen werden, sind die analogen Sensoren über drei ADC-Kanäle an den ATmega 1284p angeschlossen. Mit dem Versuchsaufbau kann so jeweils ein analoger und ein digitaler Sensor gleichzeitig abgefragt werden, wobei die analogen Sensoren auf der einen Seite des Auslegers aufgesteckt werden, während die digitalen auf der gegenüberliegenden Seite zu platzieren sind. Über jeweils einen

weiteren ADC-Kanal pro Sensor kann auch die Stromaufnahme mittels eines Shunts bestimmt werden.

Der Plattenspieler ist in der Lage, in vier verschiedenen Geschwindigkeiten zu drehen; 16 2/3, 33 1/3, 45 und 78 min^{-1}. Um das Intervall zwischen 0 und 20 m/s² (also 0 bis $\sim 2g$) abdecken zu können, wurde ein insgesamt 30 cm langer Ausleger benötigt. Die Geschwindigkeit des Plattentellers wurde mithilfe einer Lichtschranke bestimmt, sodass in die späteren Berechnungen jeweils die tatsächliche Umdrehungsgeschwindigkeit einfließt.

5.2.3 Evaluation

Alle Versuche wurden unter Laborbedingungen durchgeführt, bei konstanter Temperatur von 20 °C und 52.2° nördlicher Breite. Jeder Versuch wurde mit 5 identischen Sensoren desselben Typs durchgeführt, um produktionsbedingte Abweichungen bestimmen zu können.

Linearität

Um zu bestimmen, in wieweit die Messwerte der einzelnen Beschleunigungssensoren linear sind, wurden 30 Sensoren (sechs Sensoren, jeweils fünf Exemplare) zu 25 verschiedenen definierten Messpunkten aufgenommen, wobei sich ein Messpunkt aus der Kombination von Umdrehungsgeschwindigkeit und Abstand zum Mittelpunkt definiert. Um das System im eingeschwungenen Zustand zu erfassen, wurde jede Messung für mindestens 30 s durchgeführt. Vor jedem Versuch wurden die Sensoren über die Erdbeschleunigung (9.81 m/s²) kalibriert, um einen eventuellen Offset bestimmen zu können.

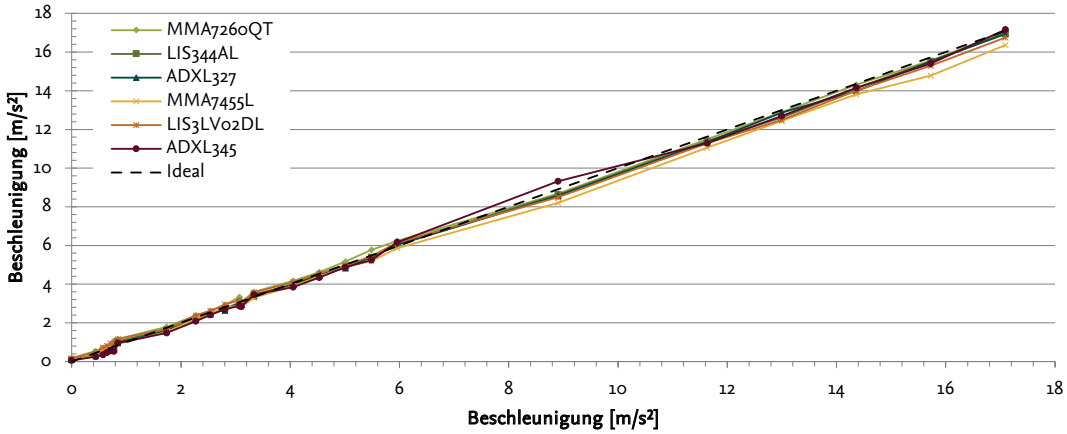


Abbildung 5.2: Die gemessene Beschleunigung der evaluierten Beschleunigungssensoren im Vergleich zum Ideal.

In Abbildung 5.2 sind die gemessenen Werte im Vergleich zum erwarteten Ideal aufgetragen. Es ist zu erkennen, dass sich keiner der Sensoren wirklich ideal verhält, jedoch auch keine schwerwiegenden Abweichungen vorhanden sind. In Tabelle 5.2 ist die Zusammenfassung der Vergleiche der gemessenen Werte mit den erwarteten Werten abzulesen. Anhand der Korrelationskoeffizienten ist zu erkennen, dass alle Sensoren relativ genau arbeiten. Die Standardabweichung wurde für zwei Intervalle getrennt bestimmt, da die Datenqualität für sehr geringe Beschleunigungen ($\leq$2 m/s^2) bei allen Sensoren nicht gut war. Ab 2 m/s^2 liegen jedoch bei allen Sensoren die Werte im akzeptablen Bereich, während darunter häufige Messfehler zu beobachten sind.

Zusammenfassend lässt sich sagen, dass sich alle betrachteten Sensoren linear verhalten und dass die Abweichungen zu einem absoluten Ideal vernachlässigbar sind. Gerade im Hinblick auf den geplanten Einsatzzweck ist die absolute Beschleunigung von weit geringerer Aussage als die relative.

Rauschen

Idealerweise würden Accelerometer in Ruhelage nur die konstante Beschleunigung in Richtung des Erdmittelpunkts detektieren. Für alle anderen Richtungen sollte demnach keine Beschleunigung gemessen werden.

Tabelle 5.2: Linearität der Beschleunigungssensoren: Messwerte im Vergleich zu dem erwarteten Werten.

Typ	MMA7260QT analog	MMA7455L digital	LIS344AL analog	LIS3LV02DL digital	ADXL327 analog	ADXL345 digital
Hersteller	Freescale		ST Microelectr.		Analog Devices	
r [1]	0.99977	0.99969	0.99979	0.99973	0.99979	0.99937
MAE [2]	0.1530	0.2581	0.1290	0.1909	0.1271	0.2042
$\sigma \leq 2$ [3]	18.25	15.53	11.14	17.20	12.20	27.78
$\sigma > 2$ [4]	2.96	4.07	2.46	2.97	2.27	4.08

[1] Korrelationskoeffizient
[2] Mean absolute error – Mittlerer absoluter Fehler in m/s^2
[3] Standardabweichung $\leq 2\ m/s^2$ in %
[4] Standardabweichung $> 2\ m/s^2$ in %

In der Realität lässt sich jedoch immer auch ein Rauschen detektieren. In Abbildung 5.3 sind gemessene Beschleunigungen aller evaluierter Sensoren in absoluter Ruhelage (beispielhaft für eine Achse) aufgetragen. Die Signale folgen dabei keinem Trend und die Werte sind diskret und zufällig verteilt. Das sogenannte Bit-Rauschen hat einen Teil seines Ursprungs in der A/D-Wandlung und gilt als Qualitätsindikator für diese Wandlung.

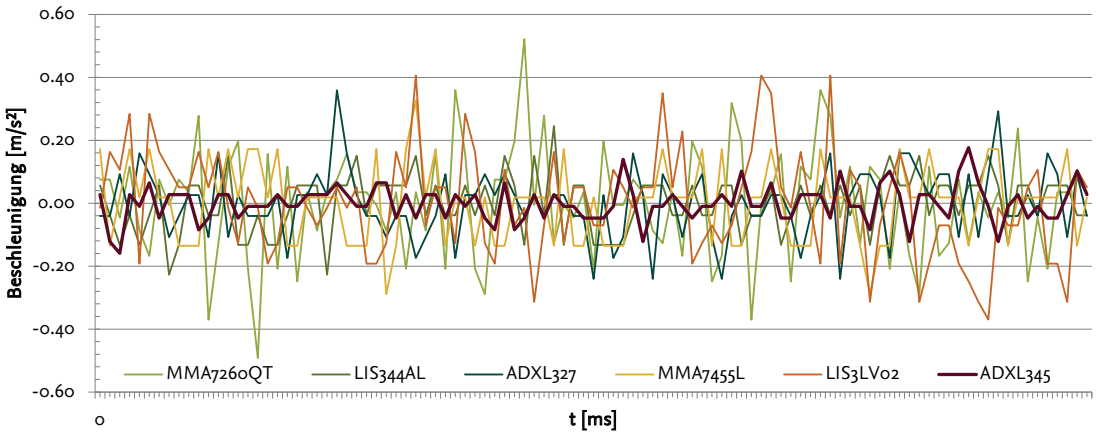


Abbildung 5.3: Die gemessene Beschleunigung der evaluierten Beschleunigungssensoren in Ruhelage.

Auf der anderen Seite spielen aber auch die individuellen Charakteristika der Sensoren eine Rolle. Wie in Abbildung 5.4 zu sehen ist, unterscheiden sich die Standardabweichungen des Bit-Rauschens auch bei den drei analogen Sensoren, welche alle mit demselben A/D-Wandler quantisiert wurden. Dort lässt sich auch erkennen, dass der digitale ADXL345 das geringste Rauschen in Ruhelage aufweist. Eine weitere Reduzierung des Rauschens lässt sich durch digitale oder analoge Filter erreichen.

Stromaufnahme

Die Stromaufnahme aller Sensoren wurde parallel zu allen durchgeführten Messungen bestimmt – die Stromaufnahme der Sensoren wurde also unter stationären und dynamischen Bedingungen evaluiert, wobei für alle Sensoren keine signifikanten Unterschiede zwischen der Stromaufnahme im Ruhezustand oder unter dem Einfluss von Bewegung festgestellt werden konnten. Das Verbrauchsmuster für einen „Timed

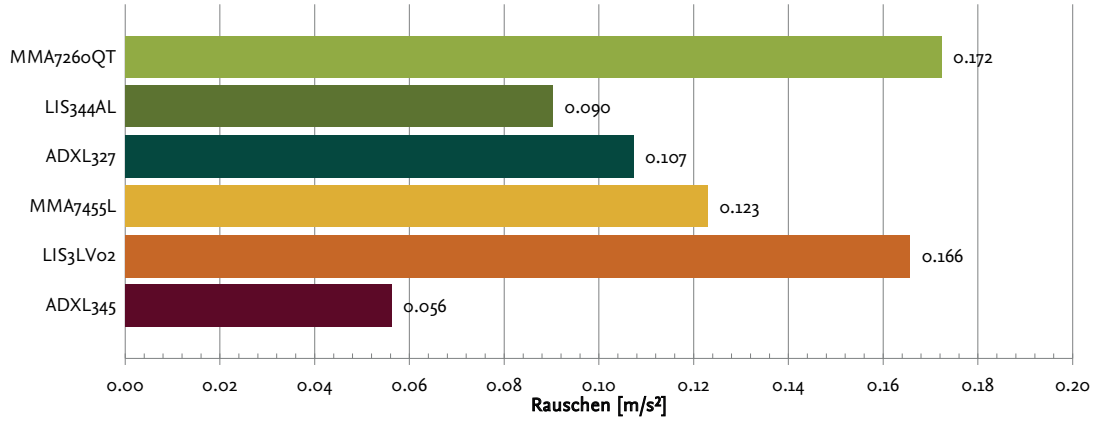


Abbildung 5.4: Die Standardabweichung der Beschleunigung der evaluierten Beschleunigungssensoren in Ruhelage.

Up&Go"-Test [27] ist in Abbildung 5.5 aufgetragen.

Bei diesen Messungen wurde allerdings jeweils nur die Stromaufnahme der einzelnen Bauteile bestimmt: Für die digitalen Sensoren heißt das, dass zusätzlich auch deren interne A/D-Wandlung mitgemessen wurde, die für die analogen Sensoren ja innerhalb des Mikrocontrollers stattfindet. Da diese A/D-Wandlung im verwendeten Mikrocontroller bei der Versuchskonfiguration $\sim 2\,\mu$A benötigt, müssten diese 2 μA jeweils zu den Messwerten der analogen Acceleroemter addiert werden.

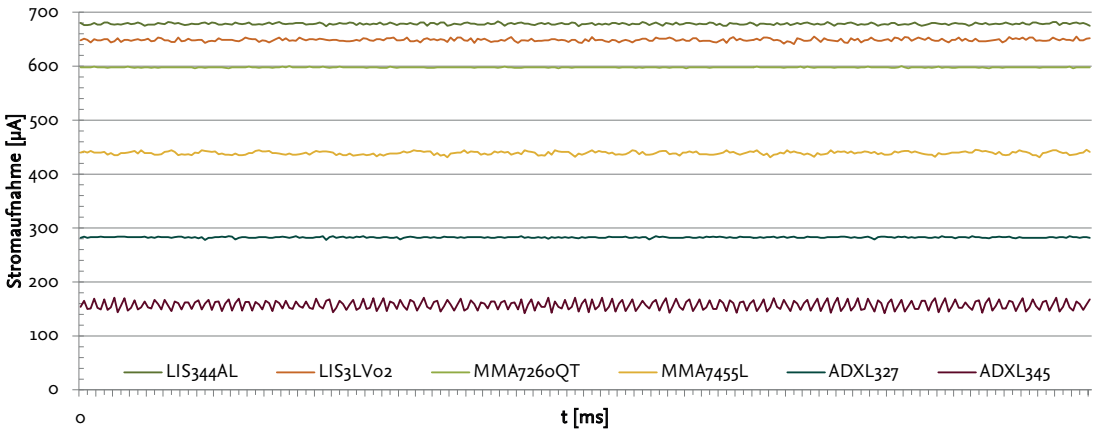


Abbildung 5.5: Die Stromaufnahme der evaluierten Beschleunigungssensoren bei einem „Timed Up&Go"-Test.

In Abbildung 5.6 ist die durchschnittliche Stromaufnahme der einzelnen Sensoren aufgetragen. Zum Vergleich ist im oberen Bereich die „typische Stromaufnahme" aus den jeweiligen Datenblättern mit aufgeführt. Hierbei fällt auf, dass es Abweichungen in beide Richtungen gibt, wobei die Stromaufnahme der analogen Sensoren immer jeweils deutlich über der Stromaufnahme ihrer digitalen Pendants liegt.

Korrelation

Der Versuchsaufbau erlaubt es, jeweils einen analogen und einen digitalen Sensor denselben Beschleunigungen zur selben Zeit auszusetzen. In Abbildung 5.7 ist beispielhaft die Korrelation der drei Achsen der beiden Sensoren ADXL345 und LIS344AL aufgetragen. Die Grafik zeigt, dass die Korrelation zwar nicht perfekt ist (in diesem Fall würden sich alle Messpunkte auf der Diagonale befinden), aber durchaus hoch ausfällt.

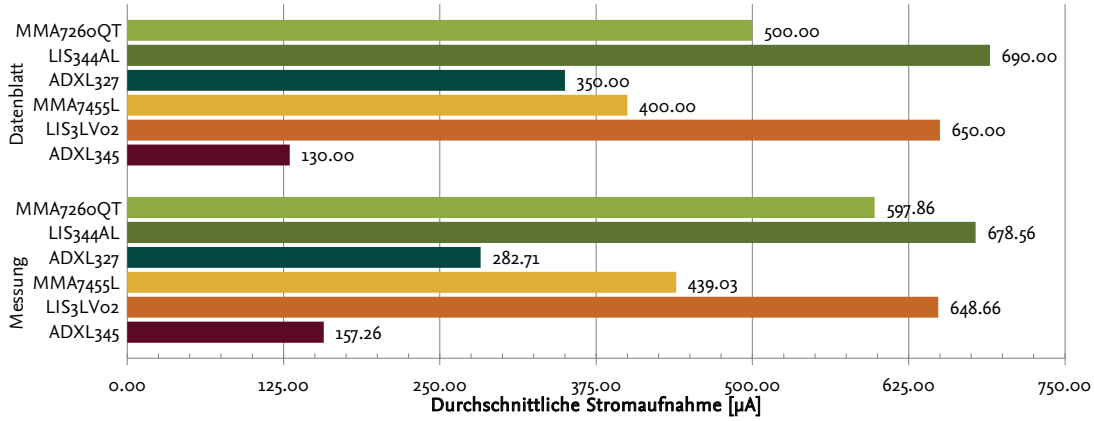
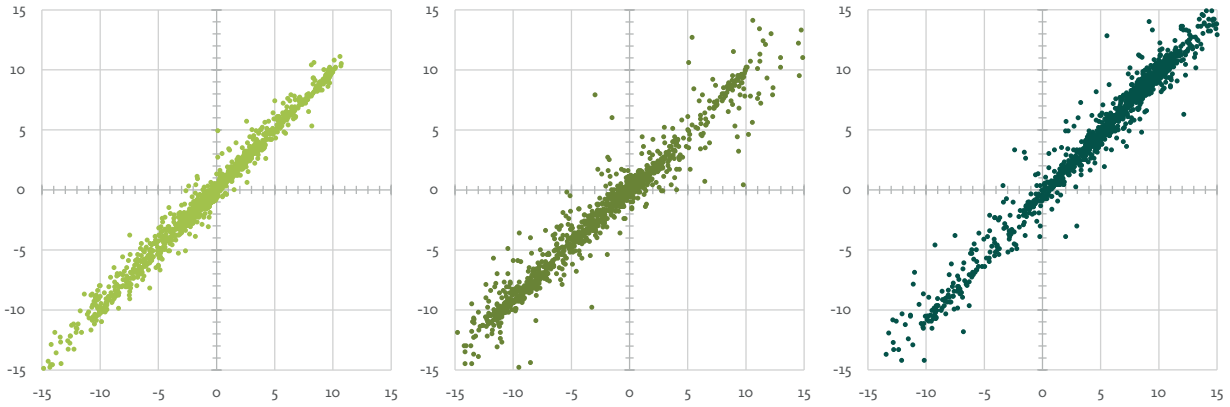


Abbildung 5.6: Die durchschnittliche Stromaufnahme der evaluierten Beschleunigungssensoren. Oben: aus Datenblättern; unten: durch Messung bestimmt.

In Tabelle 5.3 sind die Korrelationskoeffizienten für alle evaluierbaren Paare aus analogen und digitalen Sensoren aufgelistet, wobei kein Koeffizient kleiner als 0.96 ist. Diese hohe positive Korrelation legt nahe, dass die betrachteten Sensoren sehr wohl gegeneinander austauschbar sind und dass Algorithmen und Funktionalitäten, die für eines dieser Accelerometer implementiert wurden, auch auf den anderen problemlos funktionsfähig sein sollten.



(a) x-Achse (b) y-Achse (c) z-Achse

Abbildung 5.7: Korrelation zweier Accelerometer: ADXL345 (digital) und LIS344AL (analog).

5.2.4 Sensorauswahl des Accelerometers

Aus den getätigten Evaluationen geht ein klarer Favorit hervor: Das Accelerometer *ADXL345* von Analog Devices besticht durch die mit Abstand geringste Stromaufnahme, was es für mobile Anwendungen besonders interessant macht. Da es noch dazu über das geringste Bit-Rauschen verfügt und eine hohe Korrelation zu (im angestrebten Einsatzbereich häufig anzutreffenden) anderen Accelerometern aufweist, ist es die erste Wahl für den zu entwickelnden Sensorknoten. Lediglich in der Kategorie „Linearität“ gehört es nicht zu den Spitzenreitern; da aber im Anwendungsfeld der Sturz- und Aktivitätserkennung *relative* Messwerte betrachtet werden, kann die Abweichung zum absoluten Ideal vernachlässigt werden.

Tabelle 5.3: Korrelationskoeffizient r aller evaluierbaren Analog/Digital-Paare für jede Achse.

Analoger Sensor	**Digitaler Sensor**	r_x	r_y	r_z
MMA7260QT	MMA7455L	0.98793	0.98871	0.96827
MMA7260QT	LIS3LV02DL	0.99298	0.99541	0.97411
MMA7260QT	ADXL345	0.95724	0.98427	0.96453
LIS344AL	MMA7455L	0.98246	0.98670	0.98232
LIS344AL	LIS3LV02DL	0.98036	0.99580	0.99615
LIS344AL	ADXL345	0.98507	0.98806	0.98878
ADXL327	MMA7455L	0.99173	0.98973	0.98537
ADXL327	LIS3LV02DL	0.99258	0.99603	0.99539
ADXL327	ADXL345	0.99001	0.98485	0.98189

5.3 Weitere Sensoren

Wie schon in Abschnitt 2.1.1 erwähnt, werden neben Beschleunigungssensoren auch weitere Sensoren zur Aktivitäts- und Bewegungserkennung eingesetzt. Im Folgenden wird deshalb kurz auf weitere sinnvolle und benötigte Sensoren eingegangen.

Hin und wieder ist auch von Magnetometern zu lesen, welche zur Bestimmung der absoluten Lage im Raum eingesetzt werden sollen [97]. Ein Magnetometer kann Magnetfelder messen und ist im Grunde genommen ein MEMS-Kompass. Mit Hilfe des Erdmagnetfelds kann so die magnetische Nord-Süd-Ausrichtung bestimmt werden. Bei Sensoren mit 3 Achsen lässt sich so ein bodenparalleler Vektor bestimmen, der die Ausrichtung eines Körpers im Raum definiert. Was zunächst sehr interessant klingt, hat aber für die anvisierten Anwendungsfälle innerhalb von Wohnungen einen entscheidenden Nachteil: Etliche elektronische Geräte erzeugen in ihrer Umgebung ein weit stärkeres magnetisches Feld als das des Planeten. Insofern sind beim Einsatz innerhalb von Gebäuden stark fluktuierende Messwerte des magnetischen Feldes zu erwarten, weswegen auf die Integration eines Magnetometers verzichtet wird.

5.3.1 Gyroskop

Bei Ganganalysen [31][82][30][81], Bewegungsanalysen [32] und zur Sturzerkennung [23][22] kommen auch häufig Gyroskope zum Einsatz, wobei die Auswahl verfügbarer MEMS-Gyroskope sich der der Accelerometer annähert. Für die Auswahl eines passendes Gyroskops wurde jedoch auf eine ausführliche Vorabevaluation verzichtet und ein anderes Vorgehen zur Bestimmung des geeigneten Kandidaten gewählt. Da das zu entwickelnde System möglichst kostengünstig sein soll, wurden Sensoren, die pro Stück über 10 Euro kosteten, von vornherein ausgeschlossen, was die Lösungsmenge schon ziemlich eingeschränkte. Mit Apples iPhone 4 wurde zum Zeitpunkt der Konzeption der tragbaren Sensorik eines der ersten Smartphones mit einem Gyroskop ausgestattet. Das hatte zur Folge, dass das dort verbaute Gyroskop in sehr großen Stückzahlen produziert wurde, was wiederum *ein* kostengünstiges und qualitativ hochwertiges Gyroskop verfügbar machte.

Das dort eingesetzte *L3G4200D* von *STMicroelectronics* kann Drehungen um drei Achsen mit einer einstellbaren Skalierung von 250, 500 oder 2000 Grad pro Sekunde erfassen und verfügt über einen internen ADC. Die abgetasteten und quantisierten 16-Bit-Werte (pro Achse) werden mittels eines I^2C- oder SPI-Bus über einen First In–First Out (FIFO)-Puffer zur Verfügung gestellt. Es integriert einen Temperatursensor (8 Bit) und kann mit Spannungen zwischen 2.4 und 3.6 Volt betrieben werden.

5.3.2 Luftdrucksensor

Auch über den Luftdruck soll die Bewegung und die Aktivität einer Person erfasst werden können. In [24] und [84] werden Luftdrucksensoren zur Sturzerkennung eingesetzt; in [37] und [38] zur Navigation innerhalb von

Gebäuden. Dabei macht man sich zu Nutzen, dass der Luftdruck mit zunehmender Höhe abnimmt und so eine Höhenänderung der tragenden Person erkannt werden kann. Da der Luftdruck von Meereshöhe (0 m) bis 500 m im Mittel gerade einmal um 58.64 hPA also 5864 Pa abnimmt, müssen die zum Einsatz kommenden Sensoren eine hohe Auflösung im Bereich von wenigen Pascal aufweisen, um die für den Einsatz am menschlichen Körper interessanten Höhenänderungen von einigen Zentimetern detektieren zu können. Dabei ist die absolute Genauigkeit von geringerem Interesse, da in der Regel relative Änderungen erfasst werden sollen.

Die Auswahl eines geeigneten Luftdrucksensors gestaltete sich einfach, da zum Zeitpunkt der Konzeption mit *SCP1000* von *VTI Technologies* und dem *BMP085* von *Bosch Sensortec* genau zwei Kandidaten verfügbar waren, die eine benötigte Messauflösung im Bereich von 1 Pa liefern konnten. Beide wurden bereits in Studien eingesetzt. Der *SCP1000* in [24] und [84], der *BMP085* in [37] und [38].

Die Stromaufnahme im Betrieb ist beim *SCP1000* in höchster Auflösung mit 25 μA mehr als doppelt so hoch wie die des *BMP085* mit 12 μA. Bei der Stromaufnahme im Standby ist der *BMP085* mit nur 0.1 μA um eine Größenordnung besser als der *SCP1000* mit 1 μA Stromaufnahme.

Die Wahl fiel folglich auf den Sensor von Bosch Sensortec, der mit einer Fläche von 5·5 mm² und einem Spannungsbereich von 1.8 bis 3.6 Volt für den Einsatz in kleinen, mobilen und batteriebetriebenen Geräten prädestiniert ist.

5.4 Weitere Anforderungen

Neben der Sensorik sind für das in Abschnitt 2.3.3 beschriebene aggregierte Szenario und das in Kapitel 3 umrissene System, auch die weiteren Komponenten des Sensorknotens entsprechend zu dimensionieren.

5.4.1 Betriebssystem

In Abschnitt 3.1 wurde bereits erwähnt, dass es von Vorteil sein kann, ein Betriebssystem zu verwenden: So können bei einem späteren Wechsel der Hardware die einmal implementierten Funktionalitäten sinnvoll wiederverwendet werden. Die beiden bekanntesten Betriebssysteme für WSNs sind Contiki [53] und TinyOS [54]. Der häufig im Anwendungsgebiet eingesetzte Shimmer-Sensor (siehe Abschnitt 3.1.1) wird mit TinyOS ausgeliefert, weswegen eine TinyOS-Unterstützung es (ehemaligen) Shimmer-Nutzern ermöglicht, ihre Anwendungen einfach zu migrieren. In anderen Bereichen des medizinischen Monitorings wird Contiki eingesetzt [98].

Sinnvollerweise sollte der Sensorknoten also eines dieser Systeme unterstützen – idealerweise beide.

5.4.2 Prozessor und Hauptspeicher

Die gerade erwähnten Betriebssysteme wurden bereits auf unterschiedlichste Prozessorarchitekturen portiert; es liegt also nahe, auf eine Architektur zu setzen, die bereits unterstützt wird. Die im Bereich der drahtlosen Sensornetze am weitesten verbreiteten Architekturen wurden bereits in Abschnitt 3.1.1 erwähnt: MSP430 von Texas Instruments und ATmega von Atmel. Letztendlich handelt es sich um eine am Rande von Konferenzen oft diskutierte Glaubensfrage, welche Architektur „überlegen“ ist.

In Abschnitt 3.1.1 wurden bereits ein expliziter Nachteil der MSP430-Architektur erwähnt: So ist der adressierbare Programmspeicher sehr beschränkt, sodass es im GINSENG-Projekt [45] nicht möglich war, den IP-Stack um die benötigten Funktionalitäten zur Echtzeitkommunikation zu erweitern, da alleine der IP-Stack in Contiki den zur Verfügung stehenden Speicher bereits zu einem großen Teil belegte. Mit neueren Versionen seines Mikrocontrollers hat Texas Instruments an dieser Stelle aber bereits Abhilfe geschaffen.

Beide Controllertypen verfügen über zwei dedizierte Universal Synchronous/Asynchronous Receiver Transmitter (USART)-Schnittstellen. An dieser Stelle ist eine Betrachtung der Details recht aufschlussreich: Während die USART-Interfaces des MSP430 entweder als SPI, I^2C oder Universal Asynchronous Receiver Transmitter (UART) konfiguriert werden können, verfügt der ATmega über zusätzliche und separierte SPI- und I^2C-

Busse. Das ist insofern wichtig, als dass einer der USARTs in der Regel für die Kommunikation mit einem PC (z.B. über einen USB-Wandler) dauerhaft belegt ist. Beim MSP430 muss der verbleibende USART also je nach benutzter Schnittstelle (SPI, I^2C, UART) während des Betriebs umkonfiguriert werden, was zu Einbußen in der Geschwindigkeit führt.

Der zu entwickelnde Sensorknoten wird also auf der ATmega-Architektur basieren.

5.4.3 Datenspeicher und SD-Karte

Das aggregierte Szenario erfordert eine Datenspeicherung auf dem Sensorknoten, weswegen ausreichend Speicher (siehe Kapitel 4) vonnöten ist. Um auch vorhandene „Datensammelszenarien“ ohne Funkübertragung unterstützen zu können, ist es sinnvoll, diese Speicher auch austauschbar zu gestalten, weswegen nicht wie im Actigraph (Abschnitt 3.1.2) auf einen fest verbauten Speicher gesetzt wird, sondern wie beim Shimmer-Sensor ein Slot für eine microSD-Karte vorgesehen ist. Außerdem sollte eine gewisse Funktionalität zum Aufnehmen von Daten auch ohne eingelegte SD-Karte gegeben sein, was einen zusätzlichen onBoard-Flash erforderlich macht.

5.4.4 Funkinterface

In Abschnitt 4.3.2 wurde bereits ausführlich dargelegt, weswegen eine auf IEEE 802.15.4 basierende Funktechnologie zum Einsatz kommen sollte. In zahlreichen Versuchen hat sich die Funkreichweite des AVR Raven als sehr zufriedenstellend herausgestellt. Da es jedoch sinnvoll ist, auch eine Hardware-Verschlüsselung unterstützen zu können, kommt der beim AVR-Raven verwendete AT86RF230-Funktransceiver nicht infrage, da dieser dieses Feature nicht unterstützt. Mit dem Nachfolger (AT86RF231) wird allerdings eine hardwarebasierte AES-Verschlüsselung unterstützt. Die als Leiterbahn ausgeführte Antenne des AVR Raven ist (vergleichsweise) günstig und dabei sehr leistungsfähig, weswegen sich ihr Einsatz für INGA anbietet.

5.4.5 Energieversorgung

Dass ein drahtloser Sensorknoten generell energiesparend sein sollte, steht außer Frage. Dass eine beliebig lange Datenaufzeichnung und/oder -transmission mit einer begrenzten Energiequelle nicht funktionieren wird, auch. Da das Wechseln von Batterien einen größeren Aufwand darstellt, als das Laden von Akkumulatoren, sollte deshalb wie beim Shimmer-Sensor oder beim Actigraph ein wiederaufladbarer Akku verwendet werden können. Dass dabei zum Aufladen ein proprietäres Kabel oder gar eine teure Ladestation zum Einsatz kommt, steht außer Frage. Außerdem hat sich für den Einsatz am PC eine direkte Spannungsversorgung über USB als sinnvoll und praktisch erwiesen.

5.4.6 Handhabbarkeit

Separate Programmiergeräte sind zum einen ein weiterer Kostenfaktor, zum anderen auch unpraktisch, wenn es darum geht, schnell (z.B. während eines Feldversuchs) etwas zu ändern und eine neue Version der Firmware aufzuspielen. Eine komplette Programmierbarkeit über eine allgemein verfügbare Schnittstelle wie USB bietet sich deshalb an.

Für Erweiterungen jedweder Art ist es von Vorteil, wenn ein komfortabler Zugriff auf die Ports und Pins des Mikrocontrollers besteht. Kleine und proprietäre Steckverbinder sind zwar in der Regel platzsparend und machen einen professionellen Eindruck, wenn es aber darum geht, eigene Erweiterungen unkompliziert anbinden zu können, sind sie eher ein Hindernis: Unabhängig von ihrer Verfügbarkeit werden sie zumindest zusätzliches Geld kosten und aufgrund ihrer hochintegrierten Bauweise werden sie niemals so einfach zu handhaben sein wie ein 2.54 mm-Lochraster.

5.5 Architektur von INGA

Aus den erläuterten Anforderungen und Prämissen ist die Architektur von INGA entstanden, die in Abbildung 5.8 dargestellt ist. Die Basis bildet dabei ein Atmel ATmega 1284p Mikrocontroller, der über 128 kByte Flash, 16 kByte RAM und 4 kByte EEPROM verfügt. Dieser Mikrocontroller wurde bereits im AVR Raven verbaut und es existieren Portierungen für Contiki und TinyOS, was eine spätere Unterstützung der gesamten Plattform vereinfacht. Dadurch, dass alle Kommunikationsbusse separiert sind, wird ein hohes Maß an Robustheit erreicht, da fehlerhafte und falsch programmierte Bauteile primär nur den Bus beeinflussen, an dem sie angeschlossen sind, und nicht das gesamte System.

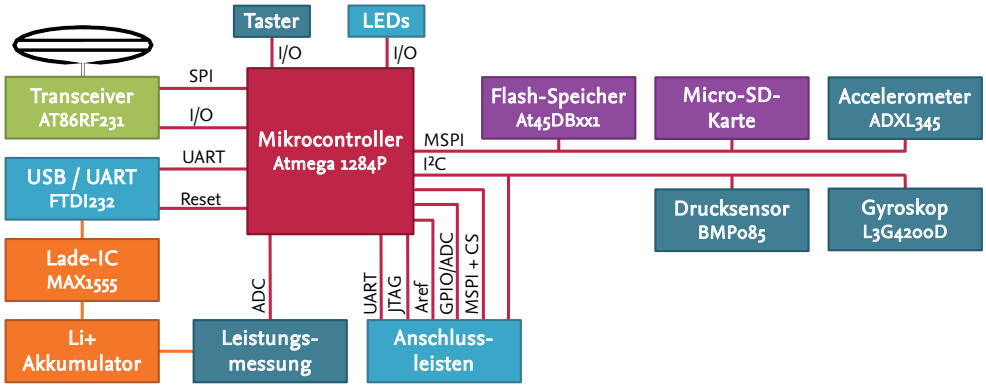


Abbildung 5.8: Blockschaltbild: Die Architektur von INGA mit den wichtigsten Bausteinen

5.5.1 USB-Schnittstelle

Die USB-Schnittstelle wurde über einen FTDI232 UART-USB-Wandler realisiert, der an den ersten USART des Mikrocontrollers angeschlossen ist. Dieser Wandler ist nur dann aktiv, wenn der USB-Port auch mit einem Host verbunden ist; in diesem Fall fungiert der USB-Port auch als Spannungsquelle für den gesamten Knoten – im kabellosen Betrieb ist er spannungslos. Der USB-Chip wird von den meisten aktuellen PC-Betriebssystemen ohne zusätzliche Treiberinstallation unterstützt.

5.5.2 Laderegler und Energieüberwachung

Wenn der USB-Port mit einer Spannungsquelle verbunden ist, wird ein an INGA angeschlossener Lithium-Polymer-Akkumulator über einen MAX1555-Laderegler geladen. Die Batteriespannung kann über einen Spannungsteiler, der an einem Kanal des im Mikrocontroller integrierten A/D-Wandlers angeschlossen ist, vom System selbst gemessen werden. Über einen Shunt, der über den Messverstärker MAX4372F mit einem weiteren A/D-Wandler-Kanal verbunden ist, lässt sich auch die momentane Stromaufnahme messen. Die Systemspannung von 3.3 Volt wird über einen linearen „Low-Dropout“-Spannungsregler des Typs MAX8881 bereitgestellt.

5.5.3 IEEE 802.15.4 Funktransceiver AT86RF231

Im Gegensatz zu seinem auf dem AVR-Raven verbauten Vorgänger (RF230) bietet dieser Funktransceiver auch die Möglichkeit der Hardware-AES-Verschlüsselung. Der Transceiver ist über einen Hardware-SPI an den Mikrocontroller angebunden, wobei er die einzige Komponente an diesem Bus ist. Die aus den Design-Richtlinien von Atmel abgeleitete PCB-Antenne bildet einen gefalteten Dipol.

5.5.4 MSPI-Bus

Über den zweiten USART ist ein Master-SPI-Bus realisiert worden, an den weitere Komponenten angeschlossen werden können. Auf diese Weise gibt es keine Beeinflussung zwischen dem Funktransceiver und den anderen SPI-Geräten. Über einen Demultiplexer wurde mit drei Port-Pins ein Chip-Select-Signal für bis zu sieben SPI-Geräte realisiert. Neben dem in Abschnitt 5.2 ausgewählten Accelerometer sind hier Flash-Speicher und SD-Karte angebunden:

Serieller Flash – Atmel AT45DBxx1

INGA kann mit einem der seriellen Flash-Bausteine AT45DB081 (8 MBit), AT45DB161 (16 MBit) oder AT45DB321 (32 MBit) bestückt werden. Diese Flash-Bausteine verfügen über einen 2-fachen Puffer, was schnelle Zugriffe ermöglicht, da ein Puffer über den SPI kommunizieren kann, während der jeweils andere für (vermeintlich länger dauernde) Schreib- oder Lesezugriffe auf den Flash-Speicher zur Verfügung steht. Dieser Aspekt wird in der Evaluation in Abschnitt 5.7 genauer untersucht.

MicroSD-Karte

Die SD-Spezifikationen sind nicht in DIN oder ISO standardisiert; vielmehr stehen sie nur zahlenden Lizenznehmern zur Verfügung. Über einen kompatiblen (aber relativ langsamen) SPI-Modus, den die meisten SD-Karten unterstützen, lassen sich diese aber dennoch kostengünstig mit einem Mikrocontroller und ohne einen speziellen Chip nutzen. Beim Betrieb von SD-Karten am SPI gibt es jedoch mehrere Fallstricke: Zum einen benötigen die Karten relativ viel Strom – je nach Kartentyp und Kartenhersteller sind 45 mA beim Schreiben keine Seltenheit. Auch im Leerlauf ist die Stromaufnahme nicht zu vernachlässigen; allerdings führt das Kappen der Versorgungsspannung in den meisten Fällen nicht zum gewünschten Effekt, da die Karte auch über die Signalleitungen Strom ziehen kann. Zum anderen verhindert das SPI-Protokoll der SD-Karte das Anschließen weiterer Geräte an den SPI-Bus, da es bestimmte Signale auf der Taktleitung (SCK) ohne gezogenen Chip-Select voraussetzt. Mit weiteren Geräten am selben Bus können so undefinierte Zustände auftreten, welche wiederum für den gesamten Bus problematisch sind.

Aus diesen Gründen können in INGAs Design alle Leitungen der SD-Karte mit einem Tri-State-Buffer vom Bus getrennt werden, was dazu führt, dass durch die SD-Karte auch (fast) keine zusätzliche Stromaufnahme verursacht wird, solange sie nicht aktiv in Benutzung ist.

5.5.5 I^2C-Bus

An den I^2C-Bus sind das in Abschnitt 5.3.1 beschriebene Gyroskop und der in Abschnitt 5.3.2 beschriebene Luftdrucksensor angebunden. Außerdem ist auch dieser Bus über einen leicht zugänglichen Pin-Header herausgeführt.

5.5.6 Bootloader

Über einen prozessorinternen Bootloader wird das Programmieren über USB ermöglicht. Dazu wird Atmels AVR109-Protokoll verwendet, welches auch kompatibel mit AVRDUDE[1] ist, was wiederum für eine Vielzahl von PC-Betriebssystemen verfügbar ist. Mit den von FTDI für Windows, Mac und Linux zur Verfügung gestellten erweiterten Treibern für den FTDI232 ist auch ein automatischer Reset des Mikrocontrollers möglich. Auf diese Weise können auch mehrere Sensorknoten auf einmal und ohne weitere Nutzerinteraktion programmiert werden. Durch die Anpassung der Controllertaktrate während des Bootloader-Betriebs kann beispielsweise Contiki in weniger als 5 Sekunden über USB aufgespielt werden.

Der Bootloader bildet außerdem die Basis für das später zu implementierende Programmieren über die Funkschnittstelle, was zumindest schon als Proof-of-Concept zuverlässig funktioniert.

[1] `http://www.nongnu.org/avrdude/`

5.6 Fertigung von INGA

Aufbauend auf der Konzeption und der Architektur wurde INGA dann komplett aufgebaut. Die eingangs aufgelisteten Kriterien wurden dabei wie folgt erfüllt:

- **State of the Art:** Mit dem ATmega 1284p wurde ein aktueller Mikrocontroller mit geringer Stromaufnahme ausgewählt. Alle angebundenen Sensoren wurden zuvor evaluiert und eignen sich für die angestrebten Anwendungsfälle.
- **Erweiterbarkeit & Handhabbarkeit:** Jeder ungenutzte Port und alle Kommunikationsbusse wurden für spätere Erweiterungen über ein 2.54 mm-Pin-Raster herausgeführt; inklusive SPI, I^2C, UART und JTAG [99].
- **Einfachheit:** Das Design wurde bewusst einfach und nachvollziehbar gehalten; dabei wurden alle benötigen Komponenten (Sensoren und Speicher) angebunden.
- **Kompatibilität:** TinyOS, Contiki und die Wiselib [100] werden von Grund auf unterstützt.
- **Kosteneffizienz:** Das Design basiert auf einer zweilagigen Leiterplatte, die auch in Eigenproduktion gefertigt und bestückt werden kann. Es kommen keine Bauteile kleiner als „0402“ zum Einsatz, was sowohl eine Bestückung von Hand als auch mit günstigen Maschinen erlaubt. INGA kann komplett bleifrei aufgebaut werden und die Spezifikationen sehen keine „problematischen Materialien“ (wie Tantal-Kondensatoren) vor.
- **Open Source:** Die unterstützen Betriebssysteme stehen ohnehin unter Open-Source-Lizenz; die Pläne und Schemata für die Hardware sind auf den Projektwebseiten[2] frei zugänglich.

In Abbildung 5.9 ist die aktuelle Version 1.4 des gefertigten INGA-Sensorknotens zu sehen. Für die Fertigung der Platinen und das Bestücken der Bauteile wurde ein entsprechender Dienstleister beauftragt. Auf die Herstellungskosten wird in Abschnitt 5.7 eingegangen.

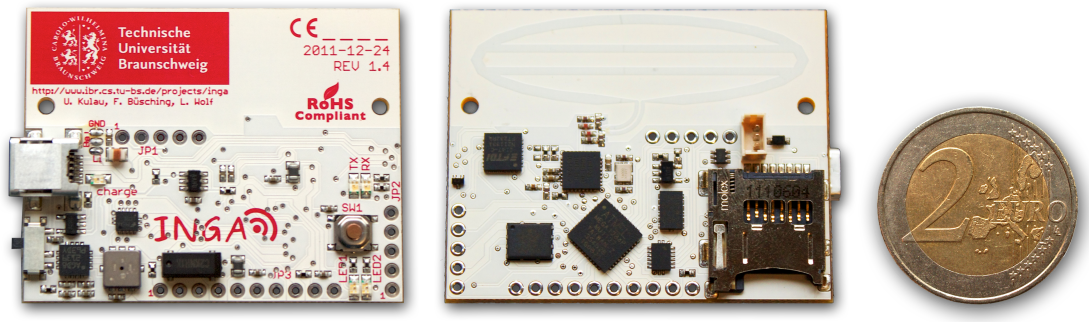


Abbildung 5.9: Vorder- und Rückseite von INGA in der Version 1.4. INGA hat eine Größe von 50 x 39 x 7 mm^3.

5.6.1 Gehäuse und Akku für INGA

Für die Inbetriebnahme und Versuche im Labor mag eine bestückte Platine ausreichend sein, für die beabsichtigten Feldversuche ist sie es nicht. Um INGA am menschlichen Körper und unter realen Bedingungen einsetzen zu können, bedarf es zum einen einer Spannungsversorgung, zum anderen aber auch eines robusten Gehäuses.

[2] http://www.ibr.cs.tu-bs.de/projects/inga

Bei der Auswahl eines geeigneten Akkus für INGA gab es zwei primäre Kriterien: Die physikalische Dimension und die Kapazität, wobei die Kapazität möglichst hoch sein sollte und die Dimension möglichst klein (bzw. an die Abmessungen von INGA angepasst). Die Fertigung eigener Akkumulatoren kam aus Kostengründen nicht infrage; auch der Einsatz einzeln erhältlicher Lithium-Polymer-Zellen erwies sich aufgrund des Preises bzw. der Qualität als wenig praktikabel. Verhältnismäßig günstig und dabei qualitativ hochwertig sind Akkus für Elektrogeräte, die in hohen Stückzahlen gebaut und verkauft werden: Akkumulatoren für Mobiltelefone gibt es in diversen Bauformen, -größen und Kapazitäten. Der Akku des *Motorola RAZR V3* hat in etwa dieselbe Grundfläche wie INGA, ist lediglich 4.5 mm dick, hat eine nominelle Kapazität von 950 mAh und kostet dabei weniger als 5 Euro.

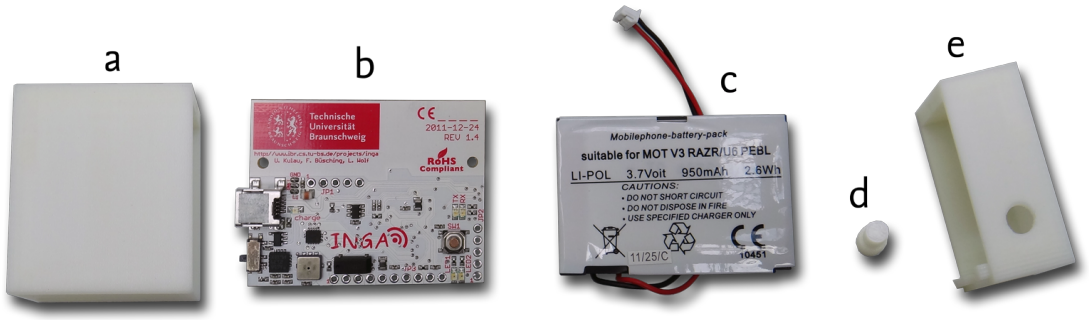


Abbildung 5.10: INGAs Gehäuse aus dem 3D-Drucker: Linke Gehäusehälfte (a), INGA (b), Akku (c), Tastknopf (d) und rechte Gehäusehälfte (e).

Ein vorhandenes und passendes Gehäuse für INGA zu finden hat sich als schwieriger herausgestellt. Mit der Technik des 3D-Drucks konnten allerdings auf einfache und kostengünstige Art und Weise Gehäuse für INGA produziert werden, die den Anforderungen an einen Feldtest genügen. In Abbildung 5.10 sind die drei Gehäuseteile mit dem Akku und INGA abgebildet.

(a) x-Achse (b) y-Achse

Abbildung 5.11: links: Gehäuse fast zusammengesetzt, rechts: komplett zusammengesetztes Gehäuse.

Abbildung 5.11 zeigt auf der linken Seite INGA im noch nicht geschlossenen Gehäuse; auf der rechten Seite ist INGA im Gehäuse abgebildet,wobei folgende Details markiert sind: Micro-USB-Port (a), Schalter (b), microSD-Slot (c) und Taster (d).

5.7 Evaluation von INGA

In diesem Abschnitt werden grundlegende Eigenschaften von INGA evaluiert. Weitere Evaluationen finden sich in den Abschnitten 6.1.4 (μDTN-Protokoll) und 6.2.4 (Verschlüsselung).

5.7.1 Funkreichweite

In einem Versuch auf dem freien Feld wurden Messungen zur Funkreichweite von INGA durchgeführt. Die Messung bestand aus einem INGA Sensorknoten und einem Atmel AVR RZ USB-Stick, der mit einem Notebook verbunden war. Es wurden vom INGA alle 20 ms UDP-Pakete mit 6 Byte Nutzlast zum Empfänger am Notebook versendet und dort ausgewertet. Parallel (aber nicht synchron) wurden dieselben Messungen auch mit einem Atmel AVR-Raven Sensorknoten als Sender durchgeführt. Die Sendeleistung beider Knoten war auf das Maximum (3 dbm) eingestellt.

Es konnte kein signifikanter Unterschied zwischen INGA und dem AVR Raven beobachtet werden: Bei beiden traten auf der gesamten Strecke vereinzelt und zufällig der Verlust einzelner Pakete auf. Ab einem Paketverlust >50 % wurde angenommen, dass das Ende der Reichweite erreicht ist und keine sinnvolle Kommunikation mehr möglich ist. Diesen Punkt erreichte INGA bei 194 m und der AVR Raven bei 219 m. Es konnte außerdem beobachtet werden, dass der Paketverlust bei dieser Schwelle innerhalb weniger Meter von fast 0 auf über 50 % angestiegen ist.

INGAs Funkschnittinterface funktioniert also und ist vergleichbar mit dem des AVR Raven.

5.7.2 Stromaufnahme

Eine geringe Stromaufnahme ist essentiell für drahtlose Sensorknoten, wobei jedoch die Stromaufnahme immer von den jeweiligen Aufgaben des Sensorknotens abhängt. Erfahrungsgemäß braucht das Senden und Empfangen von Daten vergleichsweise hohe Ströme. Das Schreiben auf eine SD-Karte kann aber im Einzelfall kurzfristig sogar weit mehr Strom als Senden oder Empfangen erfordern. Da eine Betrachtung aller möglichen Anwendungsfälle nicht durchführbar ist, soll an dieser Stelle ein Vergleich zu einem anderen Sensorknoten dienen. Dazu wurde in diesem Fall der TMote Sky Sensorknoten herangezogen und die elektrische Ladung bei beiden für das Senden von UDP-Paketen mit 80 Byte Payload bei einer Sendeleistung von 0 dBm bestimmt.

Tabelle 5.4: Stromaufnahme bei maximaler Senderate und einer Sendeleistung von 0 dBm.

	INGA	**TMote Sky**
I_{cc}	18.69 mA	19.69 mA
Maximale Datenrate	125.98 kBit/s	90.91 kBit/s
Elektrische Ladung	0.15 mAs/kBit	0.22 mAs/kBit

Tabelle 5.4 zeigt die Ergebnisse der Messung der Stromaufnahme: Zunächst fällt auf, dass bei INGA schon die absolute Stromaufnahme um mehr als 5 % geringer ist als beim TMote Sky. Da außerdem der erzielbare Durchsatz von INGA deutlich höher als beim TMote Sky ist, ergibt sich ein deutlich geringerer Strombedarf pro übertragenem Bit; INGA braucht nur 68.18 % der vom TMote Sky benötigten Ladung zur Übertragung eines Kilobit.

Abbildung 5.12 zeigt die elektrische Ladung, die benötigt wird, um ein Byte an Nutzdaten bei unterschiedlichen Nutzlastgrößen für die Protokolle UDP und RIME zu übertragen. Bei geringen Nutzlastgrößen ist INGAs Überlegenheit offensichtlich. Aber auch bei großen Nutzlastgrößen verbraucht der TMote Sky mindestens 20 % mehr Energie. Auch bei dieser Messung war der TMote Sky mit 4 MHz getaktet, während INGA mit 8 MHz lief – also versendet INGA mehr Daten, rechnet mit höheren Taktraten und verbraucht trotzdem weniger Energie.

Eine detailliertere Betrachtung von INGAs Stromaufnahme wird in [101] angestellt; dort wird mit einer modifizierten Version von INGA auch die Versorgungsspannung dynamisch geregelt.

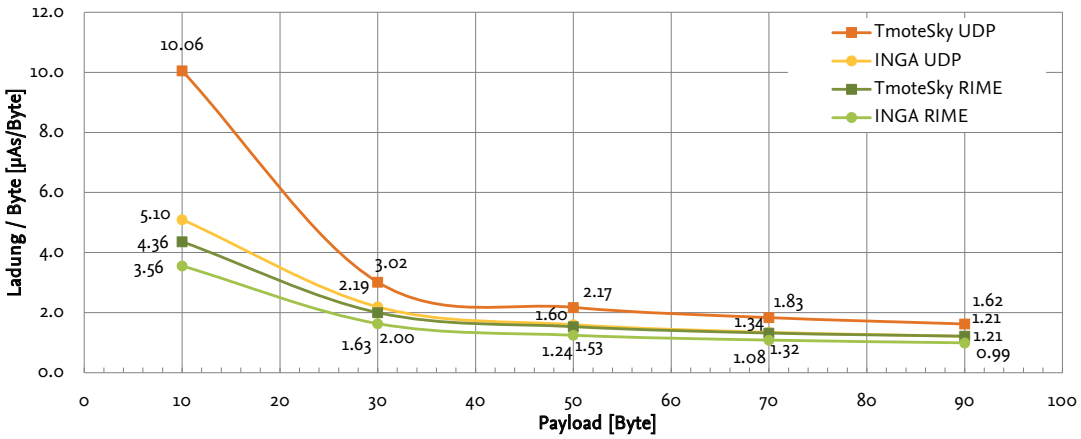


Abbildung 5.12: Die benötigte Ladung zum Versenden eines Bytes bei unterschiedlichen Transportprotokollen und verschiedenen Payloadgrößen.

5.7.3 UDP-Durchsatz

Um die allgemeine Nutzdatenrate zu bestimmen, soll an dieser Stelle zunächst das im Internet weit verbreitete UDP-Protokoll analysiert werden. Es wurde dazu die Implementierung, welche auf 6loWPAN für Contiki aufsetzt, herangezogen. Die Messungen wurden unter Laborbedingungen mit INGA und TMote Sky Sensorknoten durchgeführt, wobei auf beiden Knotentypen dieselbe Contiki-Version (2.3) lief. In Abbildung 5.13 wurde der Durchsatz über verschiedene Nutzlastgrößen (Payload) einer Datenübertragung zwischen jeweils zwei baugleichen Sensorknoten aufgetragen. Der Abstand zwischen beiden Knoten betrug einen Meter und wurde während der Versuchsdurchführung nicht verändert. Es wurden jeweils 100 Pakete versendet und die genauen Zeiten mit einem Oszilloskop bestimmt.

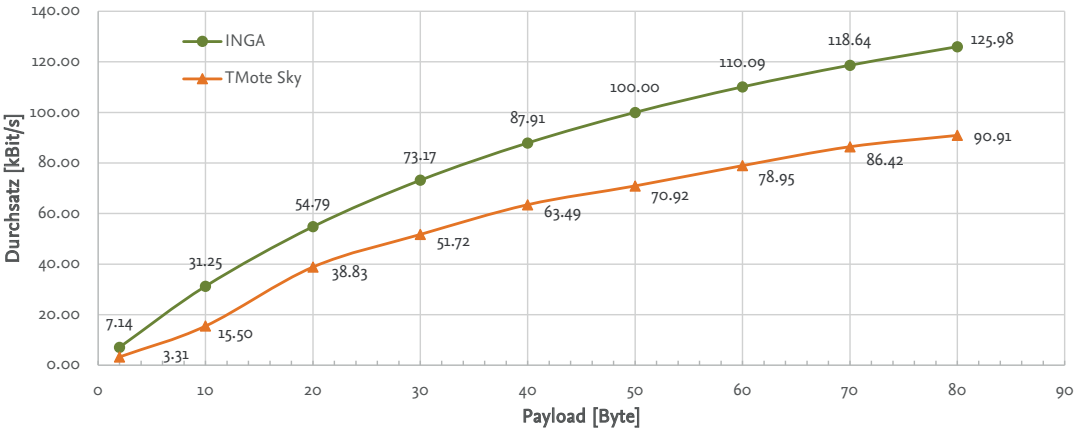


Abbildung 5.13: Der UDP-Durchsatz von TMote Sky und INGA bei unterschiedlichen Payloadgrößen.

Es ist deutlich zu erkennen, dass INGA bei allen betrachteten Payload-Größen besser abschnitt, als der TMote Sky. Bei dieser Messung trat kein Paketverlust auf, jedoch sind prinzipiell auch größere Nutzlasten denkbar, wobei an dieser Stelle der TMote-Sky nicht mehr in der Lage war, Daten in einer solchen Geschwindigkeit zu versenden, weswegen die Messung bei einer Nutzlastgröße von 80 Byte und einem daraus resultierenden maximalen UDP-Durchsatz von 125.98 kBit/s für INGA, bzw. 90.91 kBit/s für den TMote Sky abgebrochen

wurde. Mit 90 Byte Payload konnte INGA einen maximalen UDP-Durchsatz von 129.22 kBit/s erreichen.

Die großen Unterschiede zwischen den beiden Sensorknoten, die an sich eine ähnliche Leistung aufweisen sollten, ist der Tatsache geschuldet, dass in der Standardeinstellung INGA mit einer Taktfrequenz von 8 MHz läuft, der TMote Sky jedoch nur mit 4 MHz. Nichtsdestotrotz gilt für diese Standardeinstellung auch Tabelle 5.4 und INGA verbraucht trotz höherer Taktrate weniger Strom beim schnelleren Senden von mehr Daten. Im nächsten Abschnitt wird auf diese Frequenzabhängigkeit genauer eingegangen.

5.7.4 RIME-Durchsatz

RIME ist ein speziell für leistungsschwache Sensorknoten entwickelter Kommunikationsstapel. Er verfolgt einen Cross-Layer-Ansatz, d.h. oberhalb des MAC-Sublayers sind alle Funktionalitäten der darüberliegenden Schichten im RIME-Layer zusammengefasst, der wiederum eine Schnittstelle zur Anwendungsschicht zur Verfügung stellt. Da jetzt Adressierung, Routing und Transport von derselben Schicht durchgeführt werden, wird zwar die Modularität aufgegeben, jedoch ist so auch eine sehr schlanke Implementierung möglich. Wie später in Abbildung 6.9 zu sehen ist, hat RIME einen Protokolloverhead von 21 Byte – das sind zwar im Gegensatz zum gerade betrachteten UDP-over-6loWPAN nur 4 Byte mehr, jedoch ist auch die Codegröße bei RIME erheblich geringer als bei der UDP-Implementierung, was erwarten lässt, dass mit RIME eine schnellere Datenübertragung als mit UDP möglich ist.

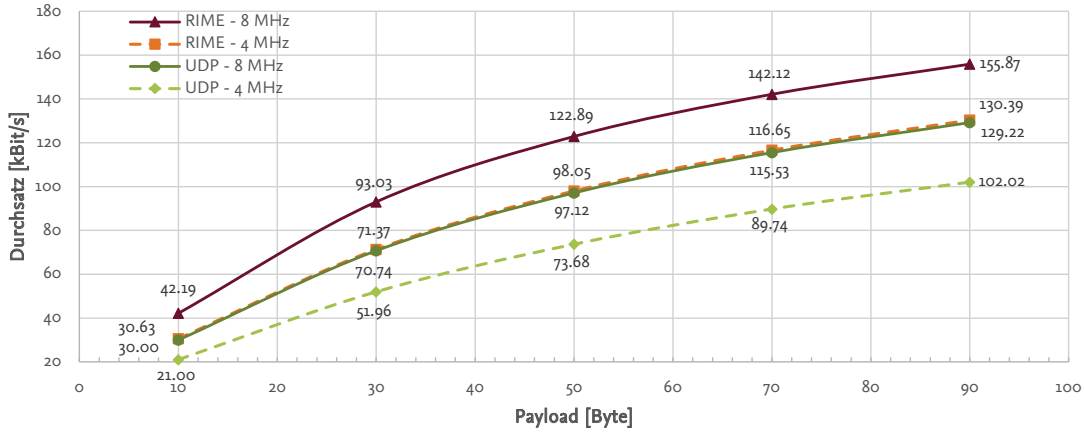


Abbildung 5.14: Der RIME- und der UDP-Durchsatz von INGA bei unterschiedlichen Payloadgrößen.

In Abbildung 5.14 wird die Effizienz von RIME sichtbar. Außerdem ist hier die Datenübertragung bei verschiedenen Taktfrequenzen von INGA zu sehen. Prinzipiell lassen sich durch den internen Oszillator zwar beliebige Taktraten einstellen, da jedoch die Timer des Betriebssystems Contiki teilweise fest auf konkrete Taktraten eingestellt sind, bietet sich für INGA nur ein Betrieb mit 4, 8 oder 16 MHz an. Da laut Datenblatt für 16 MHz die auf INGA vorhandene Prozessorspannung zu gering wäre, wird dieser Fall nicht betrachtet. Die Überlegenheit von RIME in Bezug auf den Durchsatz ist deutlich zu erkennen. Selbst bei 4 MHz Prozessortakt lassen sich (geringfügig) mehr Daten als beim 8 MHz UDP-Betrieb übertragen. Wenn man den 4 MHz UDP-Durchsatz von INGA in Abbildung 5.14 mit dem ebenfalls bei 4 MHz aufgenommenen UDP-Durchsatz des TMote Sky in Abbildung 5.13 vergleicht, ist zu erkennen, dass INGA weiterhin knapp überlegen ist, wobei auch in diesem Fall die Energieaufnahme von INGA insgesamt geringer ist.

5.7.5 Speicherdurchsatz

Für manche der betrachteten Szenarien ist auch wichtig, wie schnell die Sensorknoten in der Lage sind, empfangene Daten zu speichern. Gerade bei Netzen, die aus mehreren Knoten bestehen, von denen manche

die Daten lediglich speichern, um sie später (gegebenenfalls aggregiert) weiterzusenden, ist eine Aussage, ob und wie gut ein solches Szenario umsetzbar ist, von Belang. Dabei gilt es auch zu unterscheiden, auf welchen Speicher die Daten geschrieben werden sollen. So ist eine Speicherung im RAM sicherlich am schnellsten und effektivsten zu bewerkstelligen – allerdings ist der Arbeitsspeicher von Sensorknoten sehr begrenzt und in der Regel nicht einfach erweiterbar. INGA unterscheidet sich von anderen Sensorknoten wie dem TMote Sky durch die Anbindung der Speicher und des Funktransceivers. Während beim TMote Sky sowohl der externe Flash-Speicher als auch das Funkinterface über denselben UART angeschlossen sind, der für die unterschiedlichen Operationen jeweils anders konfiguriert werden muss, verfügt INGAs Mikrocontroller über unterschiedliche Busse für diese Aufgaben. Wenn der empfangende Knoten mit dem Schreiben der Daten auf den Speicher nicht mehr nachkommt, führt dies zu Datenverlust auf dem Funkkanal, da empfangene Pakete nicht verarbeitet und bestätigt werden können. In einer ersten Implementierung mit der Speicherung der Daten im Arbeitsspeicher der jeweiligen Knoten konnten beide Knoten bei maximaler Geschwindigkeit Daten empfangen und die Daten im RAM speichern, ohne dass es zu Paketverlust gekommen wäre – hier sind also beide Knotentypen leistungsfähig genug, um alle empfangenen Daten auch abzulegen. Bei der Verwendung von externem Speicher kommt jeweils ein gewisser Overhead hinzu, da dieser Speicher über aufwändigere Routinen angesprochen werden muss. In beiden Fällen wird mit dem externen Speicher über einen SPI-Bus kommuniziert, welcher zunächst initialisiert werden muss und über welchen dann die Steuerbefehle und Daten gesendet werden. In Abbildung 5.15 wurde deshalb die Paketzustellrate – also der Kehrwert des Paketverlusts – bei UDP-Datenverkehr am Empfänger aufgetragen, während die Daten auf den externen Flash-Speicher abgelegt werden. Als Empfänger kamen die jeweils angegebenen Knoten zum Einsatz; da der TMote Sky selbst nicht in der Lage war, höhere Datenraten zu senden, kam als Sender in beiden Fällen ein INGA-Sensorknoten zum Einsatz. Deutlich zu erkennen ist, dass der TMote Sky mit dem geteilten UART ab einem Durchsatz von 90 kBit/s beginnt, Daten zu verlieren.

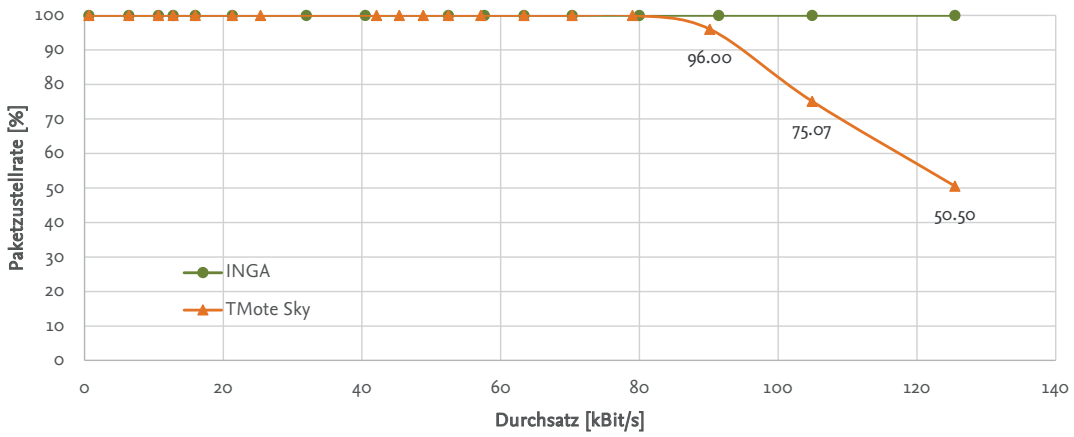


Abbildung 5.15: UDP-Paketzustellrate beim direkten Speichern auf externem SPI-Flash-Speicher.

Kritischer als das Schreiben auf den externen Flash ist das Speichern auf SD-Karten zu sehen, da diese bei der Verwendung von leistungsschwachen Mikrocontrollern meistens nur per SPI angebunden sind und nicht über (beispielsweise in Kameras oder Smartphones verwendete) proprietäre leistungsfähige Controller. So kann über SPI zwar ein kompatibles und kostenfreies, aber auch sehr langsames Interface verwendet werden. Um den Einfluss dieser langsamen Schnittstelle zu verdeutlichen, wurde in Abbildung 5.16 die Paketzustellrate beim Schreiben empfangener Daten in Abhängigkeit vom Durchsatz aufgetragen. Bis zu einem Durchsatz von ca. 40 kBit/s lassen sich alle Speicherarten ohne Einschränkung nutzen. Ab diesem Punkt steigt die Paketverlustrate beim direkten Schreiben auf die SD-Karte bis auf 100 % bei einem Durchsatz von ca. 91 kBit/s

an – die Benutzung der SD-Karte würde also unbehebbare Fehler produzieren und zu Datenverlust führen. Durch die Implementierung eines einfachen RAM-Zwischenspeichers lässt sich die Leistung beim Schreiben auf die SD-Karte erheblich steigern; selbst bei einem Durchsatz vom 80 kBit/s tritt in diesem Fall noch kein Datenverlust auf.

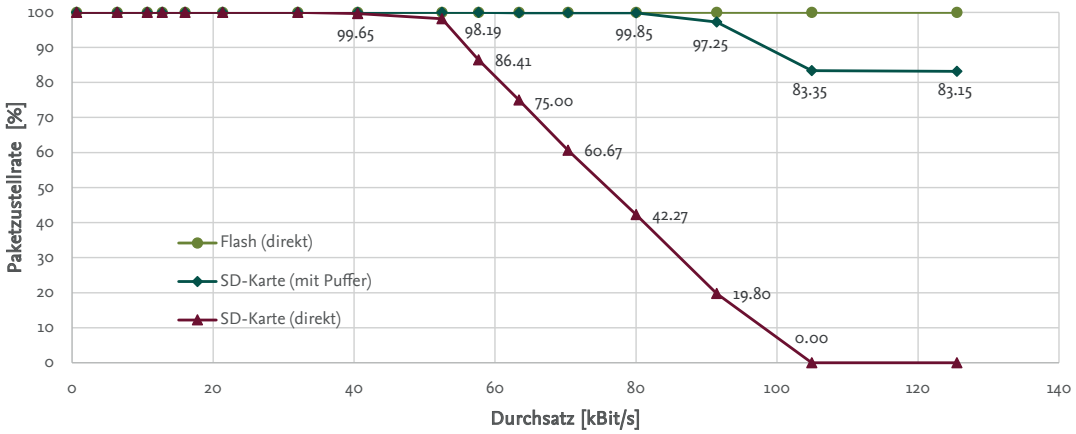


Abbildung 5.16: Paketverlust beim Schreiben auf INGAs unterschiedliche Speicher.

Wenn ein Burst-artiger Datenverkehr zu erwarten ist, also ein sehr hoher Durchsatz für einen beschränkten Zeitraum, kann es außerdem sinnvoll sein, die anfallenden Daten zunächst im externen Flash zwischenzuspeichern und dann später bei keinem oder nur geringem Datenaufkommen, umzukopieren. Das Kopieren einer Speicherseite vom externen Flash-Speicher zur SD-Karte dauert 24 ms, was zu einem Durchsatz von Flash zu SD-Karte von 21.33 kByte/s führt.

5.7.6 Kosten

Bei Konzeption und Design von INGA stand neben dem Erfüllen der technischen Anforderungen auch immer eine möglichst kostengünstige Umsetzung im Vordergrund. Die ersten beiden Prototypen waren allerdings mit 318 Euro pro Stück alles andere als günstig. Die Bestellung der ersten Charge mit insgesamt 50 Sensorknoten, kostete 82 Euro pro Sensorknoten.

In Abbildung 5.17 ist zu sehen, dass (wie zu erwarten) die Preise mit höherer Stückzahl fallen. Dabei sind die aufgeführten Preisangaben inklusive aller entstehenden Kosten für Leiterplatten, Komponenten, Bestückung, Versand und Umsatzsteuer bei einer Produktion innerhalb Deutschlands. So würden 100 Knoten je 64 Euro kosten und 200 je 61 Euro. Da die Sensoren einen großen Teil der Materialkosten ausmachen, wurden auch noch zwei weitere Fälle kalkuliert: Die Kurve „ohne Inertialsensorik“ geht von Sensorknoten ohne Accelerometer, Gyroskop und Luftdrucksensor aus. In dieser Ausstattung wären 200 Knoten bereits für 38 Euro pro Stück produzierbar. In der „minimalen Konfiguration“ wird zusätzlich auf SD-Karte und onBoard-Flash verzichtet; so lassen sich bei einer Produktion von 200 Knoten weitere 6 Euro pro Stück sparen.

5.8 Der Sensorknoten INGA – Zusammenfassung

Der Name *INGA* ist, wie oben bereits erwähnt, ein Akronym für „Inexpensive Node for General Applications“ – und das soll INGA auch sein: Die Kosten für INGA liegen weit unter denen für vergleichbare Knoten – auch wenn hier zunächst mit einem „Selbstkostenpreis“ kalkuliert wird. Eine universelle Anwendung wird unter anderem durch frei zugängliche Ports und Busse, aber auch durch eine konsequente Open-Source-Strategie unterstützt.

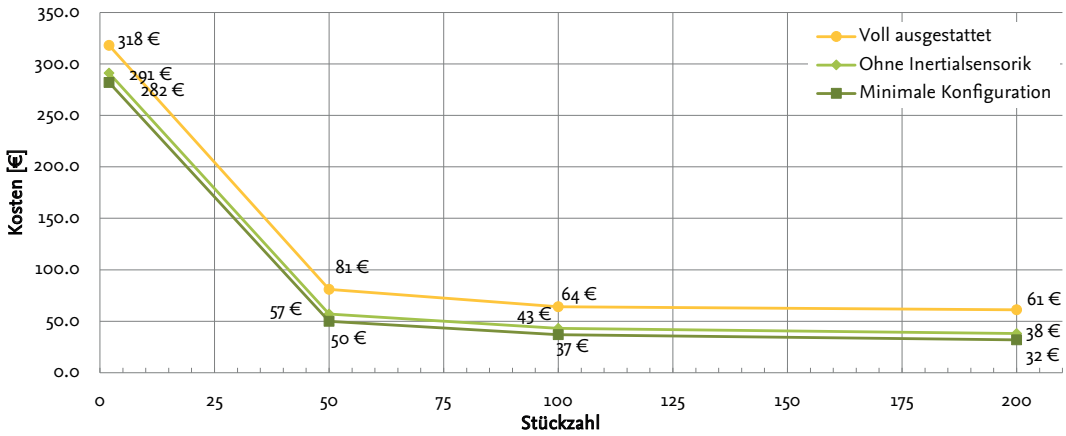


Abbildung 5.17: Herstellungskosten für INGA inklusive Bauteilen, Steuer und Versand.

INGAs Sensorik ist für die Aktivitätserkennung angepasst und ein vergleichbares Sensorset bietet kein aktuell erhältlicher Sensorknoten zu einem vergleichbaren Preis und mit vergleichbarem Volumen. Neben dem in Abschnitt 2.3.3 beschriebenen aggregierten Szenario unterstützt INGA eine Vielzahl weiterer Anwendungen. So kann dasselbe Sensorset auch zum Steuern von Quadrocoptern verwendet werden oder ein Subset auch zur Überwachung von Gebäuden [47].

In der Evaluation konnte gezeigt werden, dass INGA eine bessere Performance als der weit verbreitete TMote Sky Sensorknoten aufweist und dabei weniger Energie benötigt.

Nachdem nun eine Hardwareplattform (mit Unterstützung für zahlreiche Betriebssysteme) zum Aufnehmen, Speichern und Versenden von Aktivitätsdaten geschaffen wurde, beschäftigt sich das nächste Kapitel mit der sicheren und zuverlässigen drahtlosen Übertragung dieser Daten zu einer Senke.

6 Datenübertragung zum Gateway

„If you reveal your secrets to the wind,
you should not blame the wind for revealing them to the trees."

Kahlil Gibran – Sand and Foam, 1926

Daten, die am Körper eines Menschen aufgenommen werden, werden in der Regel an einem anderen Ort weiterverarbeitet. Zwar kann eine gewisse Funktionalität auch schon auf der körpernahen Sensorik implementiert werden, spätestens aber wenn es um die Analyse komplexer Zusammenhänge oder die Bewertung der zeitlichen Entwicklung geht, kommen leistungsfähigere Systeme oder Menschen mit entsprechender Ausbildung zum Einsatz, die diese Daten dann analysieren. Aus den Anforderungsdefinitionen in Abschnitt 2.4 ergeben sich zwei essentielle Herausforderungen an diese Datenübertragung, die beide in den Bereich der *Sicherheit* fallen:

Auf der einen Seite sollen die Daten zuverlässig übertragen werden, um eine *Betriebssicherheit* des Gesamtsystems zu gewährleisten. Ein entsprechendes Übertragungsprotokoll soll diese Aufgabe von den einzelnen Anwendungen abstrahieren, damit ähnliche Funktionalität nicht für unterschiedliche Anwendungsfälle erneut bereit gestellt werden muss. Dazu wird im folgenden Abschnitt ein unterbrechungstolerantes Übertragungsprotokoll eingeführt und vorgestellt, das diese Anforderung erfüllt. Dabei eignet sich dieses Protokoll auch implizit zur gleichzeitigen Unterstützung mehrerer Anwendungsfälle, und unterstützt so eine Online- und Offline-Auswertung der aufgenommenen Daten. Teile davon wurden vom Autor schon in [102] veröffentlicht.

Auf der anderen Seite müssen beim drahtlosen Versenden persönlicher Daten auch Aspekte der *Angriffssicherheit* beachtet werden und so insbesondere Vertraulichkeit, Integrität, Authentizität und Verbindlichkeit Berücksichtigung finden. Diese Themen werden im darauffolgenden Abschnitt 6.2 behandelt.

Schließlich gilt es auch, eventuellen Engpässen zu begegnen: In Kapitel 4 wurde bereits auf erwartbare Datenraten eingegangen. Wenn nun ein System durch die Kombination mehrere Sensoren mehr Daten erzeugt, als versendet werden können, müssen passende Mechanismen zur Datenreduktion gefunden werden. Auch aus energetischen Gesichtspunkten kann eine Datenreduktion Sinn ergeben, da das Versenden der Daten in der Regel einen nicht unerheblichen Anteil am Gesamtverbrauch des tragbaren Sensorsystems darstellt; ein reduziertes Datenaufkommen führt demnach zu weniger Übertragungen, was sich wiederum positiv auf die Lebensdauer des Gesamtsystems auswirkt. Möglichkeiten zur Datenreduktion werden dementsprechend in Abschnitt 6.3 vorgestellt und evaluiert.

6.1 Unterbrechungstolerante Netzwerke

Die Idee der Unterbrechungstoleranten Netzwerke (Disruption Tolerant Networking – DTN) [103] stammt ursprünglich aus der interplanetaren Kommunikation. Bei der Kommunikation über den Planeten Erde hinaus kann gemeinhin keine durchgehende Ende-zu-Ende Verbindung angenommen werden. Vielmehr gibt es dort (u.a. aufgrund von Verdeckungen) nur manchmal eine Verbindung zwischen zwei Kommunikationspartnern und somit hohe Unterbrechungszeiten und damit wiederum lange Verzögerungen auf der

gesamten Kommunikationsstrecke. Herkömmliche im Internet eingesetzte Transportprotokolle wie z.B. TCP oder UDP kommen hier an ihre Grenzen, da sie auf einer durchgehenden Ende-zu-Ende-Topologie aufbauen. Mit unterbrechungstoleranten Protokollen wird versucht, dem Rechnung zu tragen und dort eine verlässliche Kommunikation zu ermöglichen, wo sonst eher widrige Kommunikationsbedingungen herrschen. Erreicht wird das durch eine zuverlässige Hop-zu-Hop-Kommunikation zwischen jeweils zwei oder mehr Netzteilnehmern, sobald zwischen ihnen eine physikalische (Funk-)Verbindung besteht. Es handelt sich also um eine sogenannte „store, carry and forward“ Strategie.

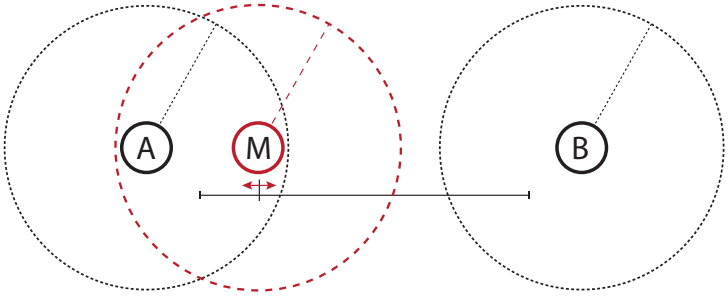


Abbildung 6.1: Grundsätzliche Funktionalität eines DTN: Knoten M bewegt sich zwischen den Kommunikationsradien der Knoten A und B. In einem DTN kann Knoten M Daten speichern, mitnehmen und weiterleiten.

Ein standardisiertes Protokoll für solche DTNs ist das Bundle-Protokoll [104], von dem es für PC-Systeme einige Implementierungen gibt, wie z.B. IBR-DTN [105]. Daten werden dabei in sogenannte Bundle verpackt und von Knoten zu Knoten weitergeleitet, wobei die Bundle eine beliebige Größe haben können. Abbildung 6.1 zeigt die einfache grundsätzliche Funktionalität: Angenommen der stationäre Knoten A möchte Daten an den nicht in seiner Funkreichweite liegenden und ebenfalls stationären Knoten B senden, so ist das mit herkömmlichen Protokollen nicht möglich. Im Fall unterbrechungstoleranter Kommunikation kann aber A B über den mobilen Knoten M erreichen, welcher die Daten physikalisch transportiert und, sobald er in Bs Funkreichweite ist, die Daten übermittelt.

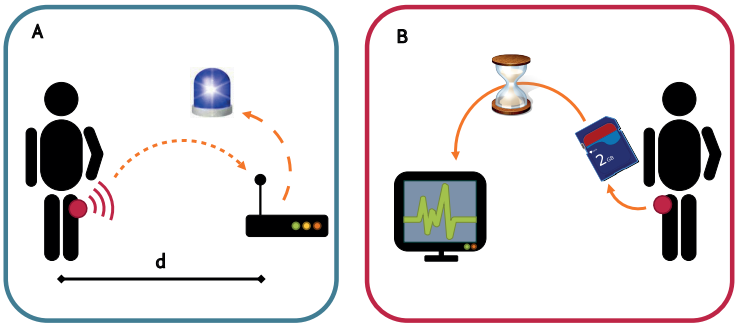


Abbildung 6.2: Zwei verschiedene Anwendungsfälle mit ähnlicher Hardware aber unterschiedlichen Anforderungen: Links (A) ist eine online-Sturzerkennung dargestellt, im rechten Fall (B) werden Aktivitätsdaten kontinuierlich aufgenommen, auf einer SD-Karte gespeichert und dann offline ausgewertet.

6.1.1 Unterstützung mehrerer Anwendungsfälle durch DTNs

In zahllosen Untersuchungen und Studien werden mit körpernaher Sensorik Vitalparameter und andere Daten aufgenommen und diese entweder direkt oder im Anschluss ausgewertet, wobei ein deutlicher Trend

zu drahtloser Datenübertragung zu erkennen ist.

In Abbildung 6.2 sind die dabei normalerweise zum Einsatz kommenden Anwendungsfälle abgebildet: In allen Studien werden die Daten früher oder später zu einer Datensenke transferiert – entweder sofort per Funk (A) oder aber nach einer gewissen Zeit, wenn z.B. die SD-Karte ausgelesen wird (B).

In vielen drahtlosen Monitoring Szenarien, bei denen z.B. ein Patient körpernahe Sensorik trägt und die aufgenommenen Daten zu einer Senke übermittelt werden sollen, ergeben sich ganz ähnliche Einschränkungen wie bei der interplanetaren Kommunikation, da auch hier oftmals keine kontinuierliche Ende-zu-Ende-Verbindung vorausgesetzt werden kann. Auf der einen Seite kann beispielsweise durch Abschattungen und unvorhersehbare Kollisionen auf dem Funkkanal die Nutzdatenrate einbrechen. Auf der anderen Seite wird beim Verlassen des Empfangsbereichs der Basis die Kommunikation vollständig abreißen. Normalerweise würde im ersten Fall die Datenqualität leiden, da einzelne Pakete verloren gehen, im zweiten Fall könnte sogar die Studie gefährdet sein, da möglicherweise wichtige Daten von bestimmten Zeiten des Tages fehlen.

In dem in Kapitel 2 skizzierten aggregierten Anwendungsfall soll davon ausgegangen werden, dass eine ältere Person, welche alleine in einer Wohnung lebt, drahtlose mobile Sensorik zur Aktivitätsüberwachung trägt. Im konkreten Fall handelt es sich dabei um ein Accelerometer, welches zum einen zur Sturzerkennung, zum anderen aber auch zur Ganganalyse und Aktivitätserkennung verwendet werden soll.

Eine Ganganalyse kann prinzipiell offline geschehen, die Daten werden also über einen längeren Zeitraum gesammelt und später an einem PC ausgewertet. Wichtig ist hier das kontinuierliche Aufzeichnen der Daten – unabhängig vom Aufenthaltsort der Person. Zurzeit werden die Daten dazu auf eine SD-Karte gespeichert, welche dann nach mehreren Tagen ausgelesen, bzw. durch eine neue ersetzt wird. Die Daten zur Sturzerkennung hingegen sollen direkt übermittelt werden, damit so zeitnah und automatisch Hilfe gerufen werden kann; die Sturzerkennung soll dabei einzig innerhalb der Wohnung erfolgen.

Da für Sturzerkennung und Ganganalyse teilweise dieselben Sensoren eingesetzt werden und somit dieselben Messwerte sowohl online also auch offline benötigt werden, bietet sich hier eine unterbrechungstolerante Übertragung an: Wenn ohnehin schon dieselbe Sensorik zur Aktivitätsaufzeichnung und zur Sturzerkennung verwendet wird, dann kann auch die Datenübertragung vereinheitlicht werden. Durch ein entsprechendes DTN-Protokoll kann diese Funktionalität für die Anwendungen transparent bereitgestellt werden. Solange sich der drahtlose Sensorknoten innerhalb der Funkreichweite eines zentralen Rechners innerhalb der Wohnung befindet, werden alle Daten auch direkt übermittelt -– sobald die Funkreichweite verlassen wird oder der Funkkanal gestört ist, werden die Daten aufgezeichnet. Bei wiederhergestellter Funkverbindung werden dann die aufgezeichneten Daten übermittelt, bzw. synchronisiert. Dass damit auch das lästige Wechseln von SD-Karten wegfällt, weil ja alle Daten an den zentralen Rechner übermittelt werden, kann als zusätzlicher Komfortgewinn angesehen werden.

In Abbildung 6.3 sind also beide Szenarien miteinander kombiniert. Auf diese Weise sind mit derselben Hardware und bei Verwendung desselben Kommunikationsprotokolls beide zunächst gegensätzlich erscheinenden Anwendungsfälle kombinierbar.

Den Anwendungsentwicklern sollte sich ein DTN-Protokoll idealerweise transparent präsentieren, d.h., dass die gesamte Funktionalität wie das Herstellen von Verbindungen, sowie das Übermitteln und Zwischenspeichern von Daten im Kommunikationsprotokoll stattfindet, und die Anwendung lediglich Daten an eine DTN-Schicht sendet und diese dann die weitere Behandlung verantwortet.

6.1.2 Kapazität von unterbrechungstoleranten Netzen

Um die Kapazität und Leistungsfähigkeit zu berechnen, werden an dieser Stelle zunächst einige Variablen definiert. Daten werden mit der Generatordatenrate D_{gen} erzeugt. Ausgehend von dem Fall, dass ein Accelerometer mit der in Kapitel 4 vorgestellten typischen Datenrate Daten erzeugt, wäre in diesem Fall $G_{gen} = 1500\,\text{Bit/s}$. Der Durchsatz auf der Anwendungsschicht (Application Layer Throughput; Goodput) wird mit D_{good} bezeich-

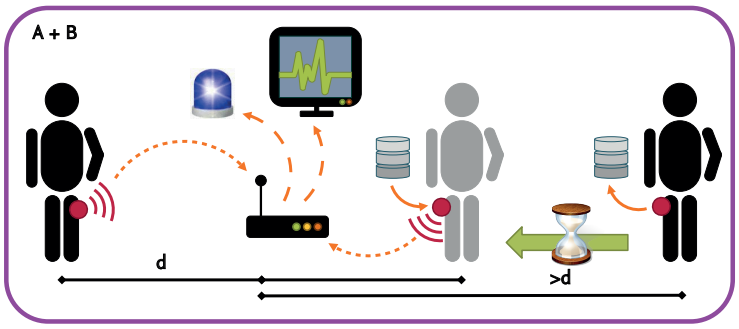


Abbildung 6.3: Kombinierter Anwendungsfall: Mit einem DTN-Protokoll können die beiden Anwendungsfälle aus Abbildung 6.2 kombiniert werden: Die online Sturzerkennung funktioniert in Funkreichweite (d) der Basisstation wie zuvor. Die Daten für die Aktivitätserkennung werden aufgezeichnet solange die tragbare Sensorik außerhalb der Funkreichweite ($> d$) ist und übertragen, sobald die Kommunikation mit der Basisstation wiederhergestellt ist ($\leq d$).

net. Vorausgreifend soll an dieser Stelle bereits davon ausgegangen werden, dass unter Laborbedingungen (siehe Abschnitt 6.1.4) mit unserer später in diesem Kapitel beschriebenen Bundle-Protokoll-Implementierung μDTN ein Goodput von bis zu $D_{good} = 50\,\text{KBit/s}$ gemessen wurde.

Spezielle Lösung

In dem Fall, dass die Datenrate der generierten Daten dauerhaft größer als der zur Verfügung stehende Goodput wäre ($D_{gen} > D_{good}$), ist nur eine lokale Speicherung (z.B. auf einer SD-Karte) möglich, da niemals alle Daten übermittelt werden könnten. Bei $D_{gen} = D_{good}$ ist zwar eine Übermittlung per Funk möglich, jedoch würde eine Verschlechterung des Funkkanals sofort zu Datenverlust führen; ein gewisses Überangebot an Bandbreite wäre also in jedem Fall erforderlich. $D_{gen} < D_{good}$ ist der für eine Unterbrechungstoleranz interessante Fall, da hier Ausfälle abgefangen und Daten später zugestellt werden können. Welche Ausfallzeiten dabei überbrückt werden können, hängt auch von der zur Zwischenspeicherung nutzbaren Speichergröße des Sensorknotens S_{node} ab, wobei der Füllstand dieses Speichers im Folgenden mit S_{fill} bezeichnet wird und sich bei anhaltender Unterbrechung der Dauer $t_{disrupt}$ mit D_{gen} füllt, wie in Gleichung (6.1) dargestellt.

$$S_{fill} = t_{disrupt} \cdot D_{gen} \tag{6.1}$$

Nachdem nun S_{fill} mit steigender Unterbrechungszeit $t_{disrupt}$ anwächst, kann bei Wiederherstellung der Verbindung mit der Synchronisation begonnen werden. Da während der Synchronisation auch weiterhin neue Daten generiert werden, steht dafür nicht der volle Goodput zur Verfügung. Wie Gleichung (6.2) veranschaulicht, ist bei der Berechnung der Synchronisationszeit t_{sync} im Nenner entsprechend D_{gen} abzuziehen.

$$t_{sync} = \frac{S_{fill}}{D_{good} - D_{gen}} \tag{6.2}$$

Mit den Gleichungen (6.1) und (6.2) lassen sich die Inhalte für Tabelle 6.1 errechnen. Hier werden die für das Szenario erreichbaren Unterbrechungs- und Synchronisationszeiten, sowie der Speicherbedarf bei $G_{gen} = 1500\,\text{Bit/s}$ und $D_{good} = 50\,\text{KBit/s}$ für verschiedene Unterbrechungsdauern gezeigt. Es ist ersichtlich, dass kurze Unterbrechungen im Minutenbereich quasi sofort synchronisiert werden können, aber auch ein ganzer Tag lässt sich innerhalb von weniger als 45 Minuten synchronisieren. Bei größeren Unterbrechungsdauern ist darauf zu achten, dass der Speicher entsprechend dimensioniert ist; ein Jahr Unterbrechung würde zwar in 11.3 Tagen synchronisiert werden können, benötigt dann aber auch eine Speicherkapazität von knapp 6 GB, was mit heutigen SD-Karten aber durchaus erreichbar ist.

Tabelle 6.1: Unterbrechungsdauer, Synchronisationsdauer sowie Speicherbedarf bei $G_{gen} = 1500\,\text{Bit/s}$ und $D_{good} = 50\,\text{KBit/s}$.

Unterbrechung	$t_{disrupt}$	t_{sync}	S_{fill}
1 Sekunde	1 s	0.03 s	187.5 Byte
1 Minute	60 s	1.86 s	11.25 KByte
10 Minuten	600 s	18.56 s	112.50 KByte
1 Stunde	3600 s	111.34 s	675.00 KByte
8 Stunden	28 800 s	890.72 s	5.40 MByte
1 Tag	86 400 s	2672.17 s	16.20 MByte
1 Woche	604 800 s	18 705.15 s	113.40 MByte
1 Monat	2 629 743.83 s	81 332.28 s	493.07 MByte
1 Jahr	31 556 926 s	975 987.40 s	5.65 GByte

Nach Zusammenfassen von Gleichung (6.1) und (6.2) fällt die Abhängigkeit vom Speicherfüllstand weg und es ergibt sich Gleichung 6.3.

$$t_{sync} = \frac{t_{disrupt} \cdot D_{gen}}{D_{good} - D_{gen}} \tag{6.3}$$

Allgemeine Lösung

Da für die Dimensionierung von unterbrechungstoleranten Systemen letztendlich das Verhältnis von D_{gen} zu D_{good} von Bedeutung ist, wird dieses durch η substituiert (siehe Gleichung (6.4)).

$$\eta = \frac{D_{gen}}{D_{good}} \tag{6.4}$$

Für den oben betrachteten speziellen Fall gilt dementsprechend $\eta = 0.03$. Nach der Substitution ergibt sich Gleichung (6.5).

$$t_{sync} = \frac{\eta \cdot t_{disrupt}}{1 - \eta} \tag{6.5}$$

Damit lassen sich für unterschiedliche Verhältnisse zwischen erzeugter Datenrate und Durchsatz auf der Anwendungsschicht η Aussagen über die Möglichkeiten zur Unterbrechungsüberbrückung treffen. In Abbildung 6.4 ist dabei die Synchronisierungszeit für unterschiedliche Werte von η bei einer Unterbrechung von einer Stunde aufgetragen. Während der ebenfalls aufgetragene Speicherverbrauch (hier für den speziellen Fall $D_{gen} = 1500\,\text{Bit/s}$) linear steigt, zeigt η ein exponentielles Wachstum. In Abbildung 6.5 sind Synchronisierungsdauer über Unterbrechungsdauer für unterschiedliche Werte von η aufgetragen. Hier ist deutlich zu erkennen, dass bei kleinen η sehr kurze Synchronisierungszeiten möglich sind, während bei großen η die Synchronisierungszeit weit über der Unterbrechungsdauer liegen kann.

6.1.3 μDTN – Eine Bundle-Protokoll-Implementierung für WSNs

Das in den vorherigen Abschnitten beschriebene System wurde entsprechend implementiert und zunächst in [106] veröffentlicht. Nach dem vorgestellten DTN-Konzept und den Bundle-Protokoll-Spezifikationen aus [104] wurde μDTN für das Betriebssystem Contiki [53] implementiert. Die Implementierung ist kompatibel zu Contikis Netzwerkstack, aber aus Gründen der Effizienz und der angestrebten Funktionsfähigkeit auf leistungsschwachen Sensorknoten mussten zwei Kompromisse getroffen werden:

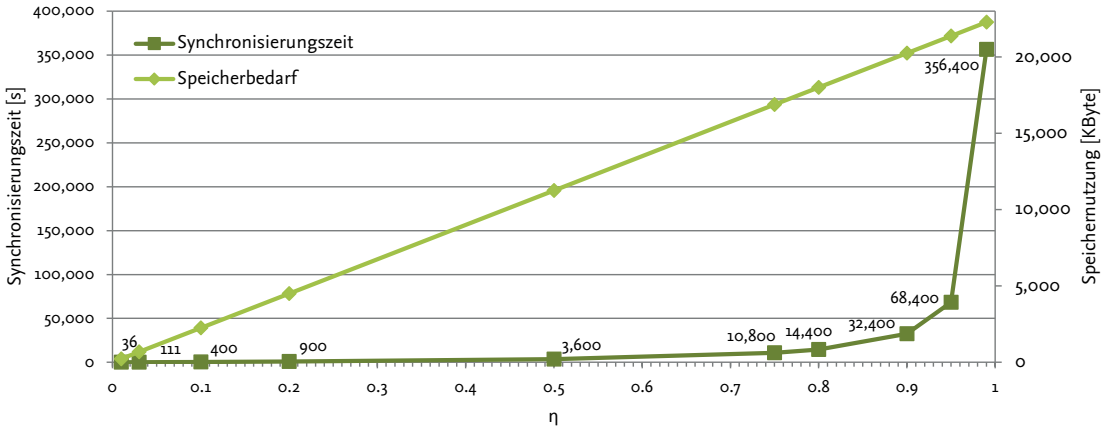


Abbildung 6.4: Synchronisierungsdauer (t_{sync} - linke Ordinate) und Speicherverbrauch (S_{fill} - rechte Ordinate) nach einer Unterbrechung von einer Stunde ($t_{disrupt} = 1h$) und unterschiedlichen η; bei konstanter Generatordatenrate $D_{gen} = 1500\,\text{Bit/s}$.

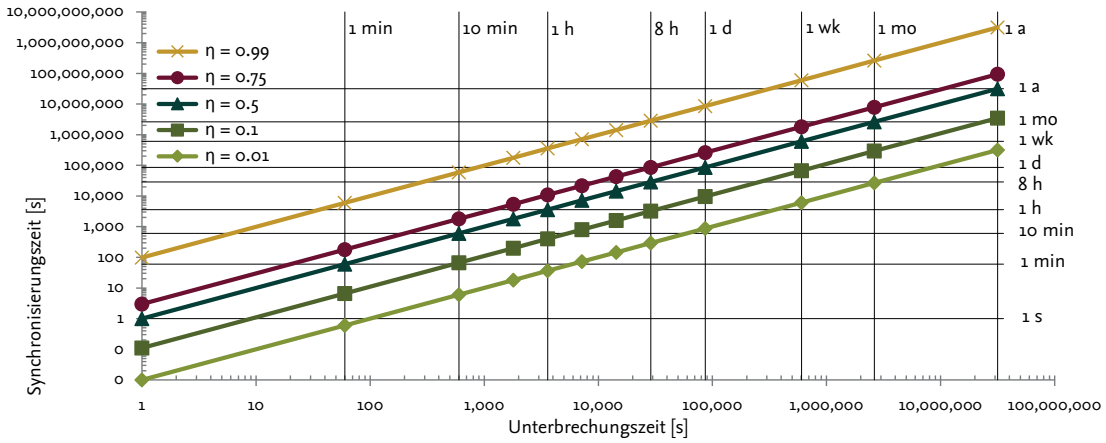


Abbildung 6.5: Logarithmische Synchronisierungsdauer über der Unterbrechungsdauer bei verschiedenen Werten für η.

- Zum einen ist nur das sogenannte *Compressed Bundle Header Encoding (CBHE)* [107] der Primärblöcke von μDTN unterstützt, was kleinere Bundle erlaubt, die einfacher zu verarbeiten sind, was wiederum zu geringerem Kommunikationsoverhead führt. Außerdem werden auch nur CBHE-konforme Adressen unterstützt (und keine Endpoint Identifiers (EIDs)).

- Zum anderen unterstützt die aktuelle Implementierung von μDTN noch keine Fragmentierung, weswegen Bundle immer in einen IEEE 802.15.4-Rahmen passen müssen. Zum jetzigen Zeitpunkt kann μDTN mit Bundle bis zu einer Größe von 94 Byte umgehen.

Wenn man diese beiden Einschränkungen berücksichtigt, ist allerdings auch eine Technologie-übergreifende Kommunikation mit anderen DTN-Implementierungen möglich, wie z.B. IBR-DTN [105], welches beispielsweise auf verbreiteten Linux-Distributionen läuft.

In Abbildung 6.6 ist die Funktionsweise von μDTN grob dargestellt. Der *Agent* stellt die zentrale Instanz dar, welche alle anderen (zumeist) modularen Komponenten eventbasiert steuert. Dabei kommuniziert der *Agent* mit der MAC-Subschicht und tauscht so Daten mit dem Funkinterface aus. Der Anwendungsschicht stellt der

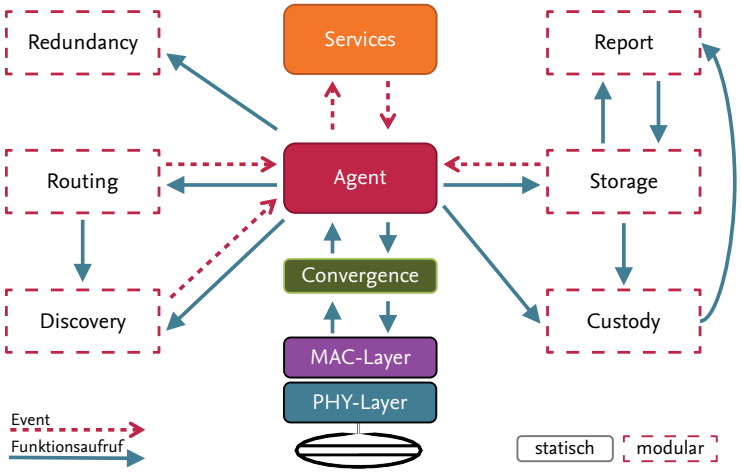


Abbildung 6.6: Der modulare Aufbau von μDTN.

Agent das Application Programming Interface (API) zur Verfügung. Hierüber werden die unterschiedlichen *Services* (Bundle-Protocol-Jargon für „Anwendungen“) abgefragt und bedient.

Über das *Discovery*-Modul können andere Knoten erkannt werden, die auch über ein IEEE 802.15.4-Funkinterface verfügen und auf demselben Funkkanal senden und empfangen. Zurzeit sind zwei unterschiedliche Discovery-Mechanismen implementiert: *IP Neighbor Discovery (IPND)* [108] und *reaktives Discovery*. Beim IPND werden bekannte Techniken aus IP-Netzwerken eingesetzt. Obwohl die Spezifikation eigentlich IP-spezifisch ist, kann das beschriebene Vorgehen auch für DTNs angewendet werden. In unserer Implementierung senden die Knoten in regelmäßigen Abständen sogenannte Beacons, die dem IPND-Format entsprechen. Der Empfänger eines solchen Beacons erkennt daraus die IEEE 802.15.4-Absenderadresse, die PAN-ID und die EID des sendenden Knotens. Im Gegensatz zu vielen anderen DTN-Implementierungen kann μDTN auch ein reaktives Discovery ausführen. Dazu sendet in diesem Fall ein Knoten nur dann ein Beacon, wenn er auch Daten zu versenden hat. Auch dieser Beacon kann dann von Nachbarknoten in Funkreichweite empfangen werden. Jeder erkannte Nachbarknoten wird in eine Liste eingetragen und dann beispielsweise über den *Agent* an das Routing-Modul gemeldet, damit gegebenenfalls eine Datenübertragung stattfinden kann.

Das *Routing*-Modul entscheidet, welche Bundle an welche Nachbarknoten gesendet werden. Die Informationen über erreichbare Nachbarn erhält es dabei vom gerade beschriebenen Discovery-Modul. Außerdem erhält das Routing-Modul beim Löschen oder Speichern eines Bundle Nachrichten, damit es auf die Situation reagieren kann und Bundle an eventuell gerade vorhandene Nachbarn zustellen kann. Anhand unterschiedlichster Routing-Algorithmen und –Strategien kann dann entschieden werden, ob und wenn ja welches Bundle an welchen Nachbarn zugestellt werden soll, wobei die Routingstrategie jedoch stark von der jeweiligen Anwendung abhängig ist. Im betrachteten Anwendungsfall mit nur zwei Kommunikationspartnern, führt das Routing-Modul konventionelles Flooding aus, d.h. jedes Mal, wenn das Discovery-Modul einen Kommunikationspartner „entdeckt“, werden bisher noch nicht übermittelte Bundle zugestellt.

Das *Storage*-Modul ist verantwortlich für das Speichern, bzw. Zwischenspeichern der einzelnen Bundle. Der Agent kann Bundle wiederum vom Storage-Modul anfordern. Mit dem Zeitstempel der Erzeugung, der Sequenznummer, dem Offset des Fragments und der EID des Erzeugers lassen sich laut [104] Bundle eindeutig identifizieren. Da aber ein 1-zu-1-Vergleich all dieser Daten auf leistungsschwacher Hardware sehr zeitaufwändig sein kann, wird intern ein eindeutiger 16-Bit-Wert generiert, der einen schnelleren Zugriff auf die jeweiligen Bundle erlaubt. Generell sind mehrere Arten der Speicheranbindung vorgesehen, die durch jeweils unterschiedliche Storage-Module realisiert werden können. Durch die verschiedenen Arten der

Speicheranbindung sind auch unterschiedliche Speichertypen benutzbar. Als Storage kommt allerdings im Augenblick nur RAM zum Einsatz.

Da die einzelnen Bundle auf unterschiedlichen Wegen zum Ziel gelangen können, kann es vorkommen, dass Bundle mit demselben Inhalt mehrfach am Empfängerknoten ankommen; das *Redundancy*-Modul entscheidet darüber, ob empfangene Bundle zu lokalen Anwendungen „durchgestellt" werden oder als Duplikat verworfen werden. Auf den ersten Blick mag das eine ziemlich triviale Aufgabe sein, welche nicht in ein eigenständiges Modul ausgelagert werden muss, jedoch ist das Vergleichen eine relativ aufwändige Prozedur, welche anwendungsorientiert optimierbar ist.

Der Empfänger eines Bundle muss nicht zwangsweise auch das Ziel eines Bundle sein. In Multi-Hop-Szenarien, in denen Bundle über mehrere Zwischenstationen gesendet werden, ist es sogar die Ausnahme, dass ein Bundle direkt zugestellt wird. Wenn Bundle also über mehrere Zwischeninstanzen versendet wird, kann ein weiterleitender Knoten die Obhut (engl. Custody) für dieses Bundle übernehmen, sofern dieses vom ursprünglichen Sender angefordert wurde. Es kann auf diese Weise eine bestätigte Übertragung von beispielsweise besonders wichtigen Bundle erfolgen. Im *Custody*-Modul werden also die Übernahme und die Abgabe von „Verantwortung" für Bundle geregelt. Außerdem kann hier die Entscheidung getroffen werden, ob der betreffende Knoten überhaupt in der Lage (bzw. „gewillt") ist, das betreffende Bundle zu verwalten, wobei hierbei auch der Speicherfüllstand oder die verbleibende Energie eine Rolle spielen können.

Über Berichte, die vom *Status-Reports*-Modul dann erzeugt werden, kann dann der Status über Übernahme und Abgabe von Verantwortung kommuniziert werden.

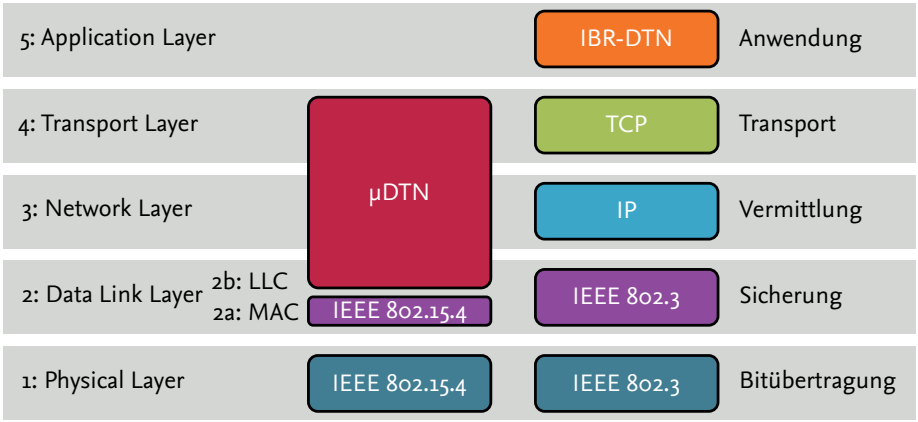


Abbildung 6.7: μDTN im Vergleich zu IBR-DTN im 5-Schichten-TCP/IP-Referenzmodell.

In Abbildung 6.7 ist μDTNs Einordnung in das 5-schichtige TCP/IP-Referenzmodell im Vergleich zu einer beispielhaften IBR-DTN Anwendung, die im Beispiel auf Ethernet (IEEE 802.3) und TCP/IP basiert, dargestellt: Durch IEEE 802.15.4 ist sowohl die Bitübertragungsschicht (1), als auch die Medienzugriffs-Subschicht (2a) definiert. Im Gegensatz zu den meisten anderen DTN-Implementierungen, die meistens als Anwendungen oberhalb der Schicht 4 laufen, baut μDTN direkt auf der IEEE 802.15.4 Medienzugriffsschicht (MAC-Sublayer) auf, was wiederum den Overhead der dazwischenliegenden Schichten spart und eine effiziente Implementierung für Mikrocontroller erlaubt. Wie in den Abbildungen 6.6 und 6.7 zu sehen, übernimmt μDTN als sogenannter *Cross-Layer*-Ansatz die Aufgaben der Netzwerkschicht (Adressierung und Routing) und der Transportschicht (transparenter und zuverlässiger Ende-zu-Ende-Datenverkehr).

Da die Bitübertragungsschicht (Physical Layer (PHY)) von IEEE 802.15.4 eine maximale PHY Service Data Unit (PSDU) Länge von 127 Byte verarbeiten kann, muss auch der IEEE 802.15.4 MAC-Rahmen in diese maximalen 127 Byte passen. In Abbildung 6.8 ist die Unterteilung des IEEE 802.15.4-MAC-Rahmens in MAC-Header (MHR),

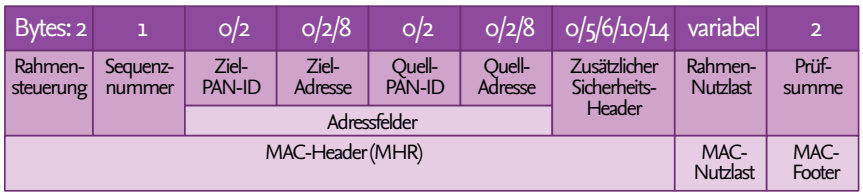


Abbildung 6.8: Der Aufbau eines IEEE 802.15.4-MAC-Rahmes.

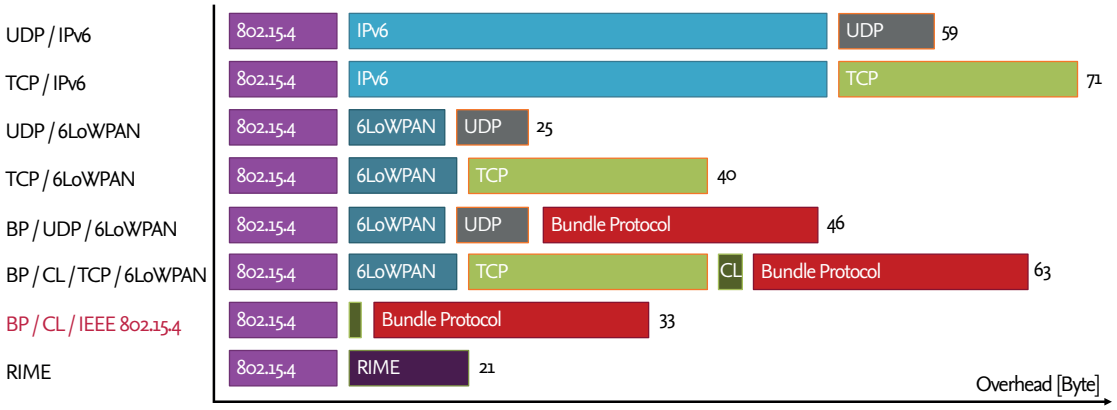


Abbildung 6.9: Overhead von vorhandenen Kommunikationsstacks im Vergleich zu möglichen Implementierungen für ein Bundle-Protokoll bei der Verwendung von IEEE 802.15.4.

acMac-Nutzlast und MAC-Footer (MFR) zu sehen. Wie zu erkennen, beträgt der minimale Overhead für den Header 3 Byte für Rahmensteuerung und Sequenznummer und im Footer konstant 2 Byte für eine Prüfsumme. Will man außerdem noch Ziele adressieren, kommen je nach Kodierung bzw. Länge der Zieladresse noch einige Byte hinzu. Ohne die Nutzung des optionalen Sicherheitsheaders, welcher bis zu weiteren 14 Byte belegen kann [109], kann der Header/Footer-Overhead bis zu 25 Byte groß werden. Das ist auch der Standardwert, der in Contiki für den MAC-Overhead angenommen wird, und auf dem alle auf 6LoWPAN [68] basierenden Protokolle aufsetzen.

Für die Implementierung von μDTN für Contiki ist der Convergence Layer (CL)-Header immer 1 Byte lang. Der restliche Overhead ergibt sich aus den Spezifikationen der einzelnen Felder und ist unter anderem in [104] beschrieben. Je nach Länge der einzelnen Felder ergibt sich für μDTN ein zusätzlicher Overhead von mindestens 18 Byte. Die Größe ergibt sich aus einer Konfiguration mit 7 Bit großen Sequenznummern und 7 Bit großen Adressen, was zu einem *Primary Bundle Block* von 15 Byte führt sowie einem 3 Byte großem *Bundle Payload Block Header* bei einer maximalen Länge des Payloads von 127 Byte. Außerdem werden mindestens 11 Byte des IEEE 802.15.4-Headers/Footers benötigt, welche im Header die Rahmensteuerung (2 Byte), eine Sequenznummer (1 Byte), Adressinformationen (mind. (6 Byte)) und im Footer eine 2 Byte große Checksumme beinhalten. Nutzbar für die Nutzdaten wären demnach noch 97 Byte.

Es kann allerdings davon ausgegangen werden, dass für größere Verzögerungen auch größere Sequenznummern sinnvoll sind; auf der anderen Seite ist auch eine größere Lebensdauer (*Lifetime*) der einzelnen Bundle wünschenswert. Das wiederum wirkt sich auf die Größe des Overheads aus, der für diese Konfiguration (ein weiteres Byte für größere Sequenznummern, zwei weitere Byte für die Lifetime) auf 21 Byte wächst, weswegen von einer für den Anwendungsfall sinnvollen Nutzdatenlänge von maximal 127-11-1-21=94 Byte ausgegangen werden kann. In Abbildung 6.9 ist dieser Overhead in Vergleich zum Overhead anderer Protokolle dargestellt.

Der unkomprimierte IPv6-Header ist beispielsweise alleine 40 Byte groß, der UDP-Header würde weitere 8 Byte einnehmen. Zusammen mit 11 Byte für Adressierung und MAC-Overhead ergäbe sich ein Gesamtoverhead von 59 Byte; das heißt, dass fast die Hälfte der nutzbaren Rahmengröße von 127 Byte verbraucht würde. Bei TCP über IPv6 wäre der Overhead noch größer. Würde man darauf aufbauend eine Bundle-Protokoll-Implementierung auf Ebene der Anwendungsschicht implementieren, stände ein sehr großer Overhead einer sehr geringen Nutzdatengröße gegenüber.

Bei der Nutzung von 6LoWPAN kann der IPv6-Overhead signifikant verringert werden. Konfiguriert man zusätzlich den 6LoWPAN-Overhead auf nötige 11 Byte (anstelle der in Contiki vorgesehenen 25 Byte) lässt sich der Gesamtoverhead für UDP über 6LoWPAN auf 25 Byte reduzieren. Eine darauf aufbauende Bundle-Protokoll-Implementierung käme hierbei ohne CL aus und würde zu einem Gesamtoverhead von 46 Byte führen. Bei TCP über 6LoWPAN hingegen würde neben dem höheren Overhead auch ein weiter CL benötigt, was zu einem theoretischen Overhead von 63 Byte führen würde.

Außerdem ist in Abbildung 6.9 noch RIME [70] als Protokoll mit sehr geringem Overhead aufgeführt. Da RIME jedoch ausschließlich für Contiki verfügbar ist, wird von einer DTN-Implementierung basierend auf RIME abgesehen.

μDTN wurde also zur Vermeidung des Kommunikationsoverheads direkt oberhalb des MAC-Sublayers implementiert. Da mit geringerem Overhead der Durchsatz steigt, hat das positive Auswirkungen auf die Performance. Negative Auswirkungen auf die Performance ergeben sich durch die Kodierung der Header-Felder als Self-Delimiting Numeric Values (SDNVs), also sich selbst beschränkende numerische Werte. Der Vorteil, dass die Felder eine variable Länge haben können, wird durch aufwändige Shift-Operationen erkauft. Für eine Implementierung auf leistungsschwachen Mikrocontrollern wären eigentlich feste Größen besser geeignet, das würde jedoch den Bundle-Protokoll (BP)-Spezifikationen widersprechen und so die Interoperabilität mit BP-Implementierungen der PC-Welt gefährden.

Eine genauere Beschreibung von μDTN findet sich in den Publikationen [106], [110] und [111]. Als nächstes soll nun die Leistungsfähigkeit von μDTN evaluiert werden.

6.1.4 Evaluation von μDTN auf INGA

Die Evaluation findet dazu in mehreren Schritten statt. Zunächst soll die reine Funktionalität im Rahmen von Labortests bestimmt werden. Als nächstes wird die Leistungsfähigkeit von μDTN dann im Rahmen des Gesamtsystems und den realen Bedingungen eines Feldtests betrachtet und auf Schwachstellen und Fehler untersucht.

Laborversuche

Ziel dieser Evaluation ist zunächst die Bestimmung der grundsätzlichen Funktionalität und die Bestimmung der tatsächlich erreichbaren Nettodatenrate (Goodput).

Bei Standard-WLAN (IEEE 802.11b) liegt die Nettodatenrate mit 5.5 MBit/s nur in etwas halb so hoch, wie die nominelle Datenrate von 11 MBit [112]. Auch bei den nachfolgenden und verbesserten IEEE 802.11(a/g/n/..)-Standards liegt die Nettodatenrate – unter idealen Bedienungen – bei maximal der Hälfte der nominalen Bruttodatenrate [113]. Da die funktionsbedingte Komplexität von μDTN größer als bei anderen Protokollen ist, wird eine schlechtere Ausnutzung der zur Verfügung stehenden Kapazität erwartet.

Im Laborversuch haben zwei baugleiche Knoten einen Abstand von 1 m und die Daten werden von einem Knoten (Sender) zu einem anderen Knoten (Empfänger) übertragen. Zum Vergleich werden die Durchsätze von RIME, User Datagram Protocol (UDP) und μDTN jeweils in der gleichen Umgebung bestimmt. In den Abbildungen 6.10 und 6.11 ist jeweils die Nettodatenrate bei verschiedenen Paket- bzw. Bundlegrößen aufgetragen – und zwar für einen Prozessortakt von INGA von 8 bzw. 4 MHz. Die maximale durchschnittliche Nettodatenrate bei 8 MHz beträgt für 90 Bundle pro Sekunde 44.57 kBit/s, wobei die Standardabweichung mit

σ = 4.69 relativ hoch ist. Bei einigen Messungen konnten unter Idealbedingungen auch Werte von knapp über 50 kBit/s gemessen werden, was als erreichbarer Wert in die Kapazitätsabschätzung (Abschnitt 6.1.2) eingegangen ist. Aufgrund der Komplexität der Implementierung und der aufwändigen SDNV-Berechnungen schafft μDTN nur in etwa den halben Durchsatz von UDP. RIME hingegen schneidet mit 106.97 kBit/s noch besser ab.

Bei 4 MHz ergibt sich in Abbildung 6.11 ein ähnliches Gesamtbild, wobei die Einbußen von μDTN deutlich größer ausfallen, als bei den anderen Protokollen.

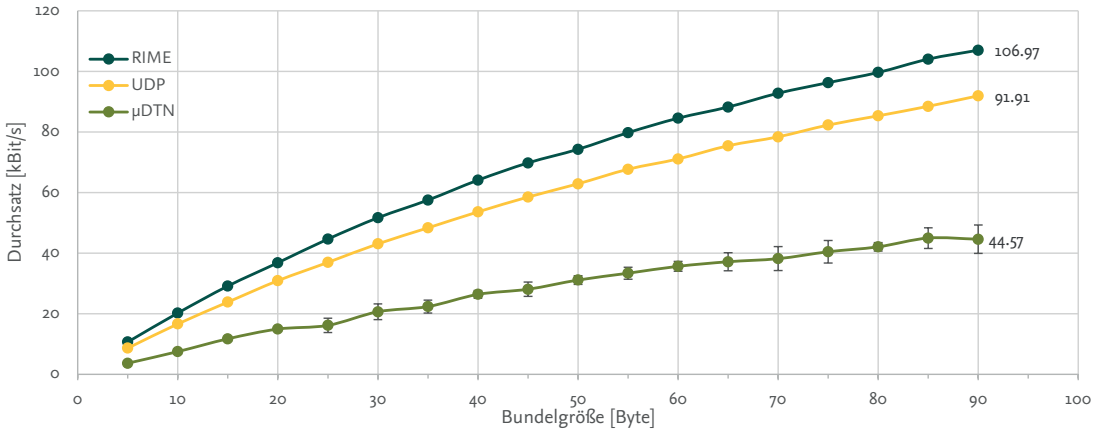


Abbildung 6.10: Der im Labor bestimmte Durchsatz der Übertragungsprotokolle RIME, UDP und μDTN bei verschiedenen Paket- bzw. Bundlegrößen und einem Prozessortakt von 8 MHz.

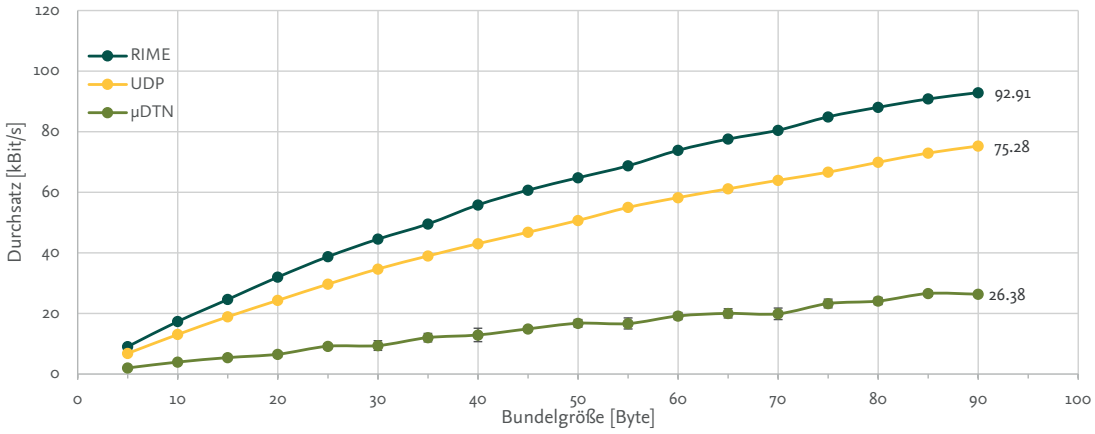


Abbildung 6.11: Der im Labor bestimmte Durchsatz der Übertragungsprotokolle RIME, UDP und μDTN bei verschiedenen Paket- bzw. Bundlegrößen und einem Prozessortakt von 4 MHz.

In den Abbildungen 6.12 und 6.13 sind die pro Sekunde gesendeten Pakete bzw. Bundle aufgetragen. Hier fällt besonders auf, dass bei 4 MHz für μDTN die Anzahl der Pakete pro Sekunde eher konstant bleibt, während sie bei den anderen Protokollen kontinuierlich sinkt. Das lässt den Schluss zu, dass hier ein statischer Overhead den Durchsatz begrenzt, der wahrscheinlich von der komplexen Berechnung der Header-Felder herrührt, welche ja unabhängig von der Nutzlast berechnet werden müssen.

Während bei allen anderen Protokollen jedoch – bei geringen Datenraten sporadisch, bei hohen Datenraten häufig (siehe dazu auch Abbildung 6.16) – Übertragungsverluste auftraten, funktionierte μDTN fehlerfrei.

Letztendlich ist für die Wahl des geeigneten Protokolls für die betrachteten Anwendungsfälle nicht ein möglichst hoher Durchsatz, sondern die Fähigkeit, mit Verzögerungen und Unterbrechungen umgehen zu können, ausschlaggebend.

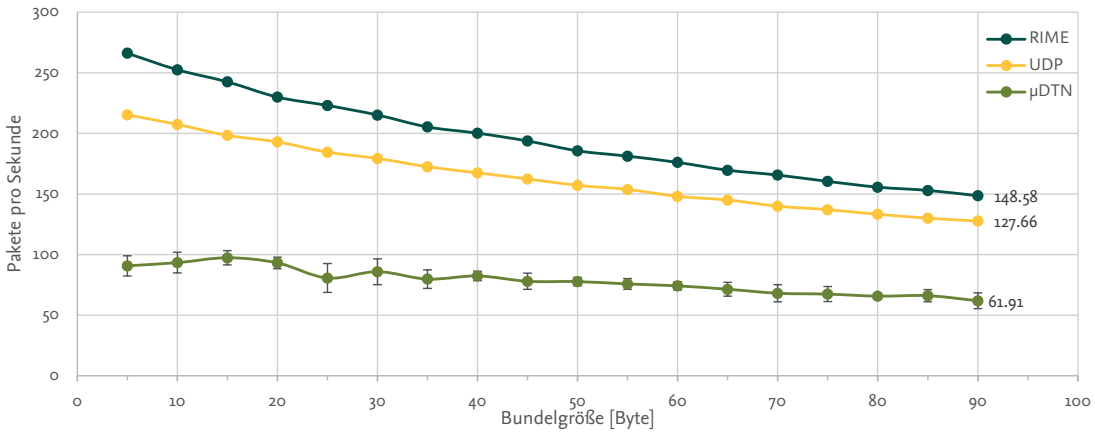


Abbildung 6.12: Die im Labor bestimmte Anzahl von Paketen bzw. Bundle pro Sekunde der Übertragungsprotokolle RIME, UDP und μDTN bei verschiedenen Paket- bzw. Bundlegrößen und einem Prozessortakt von 8 MHz.

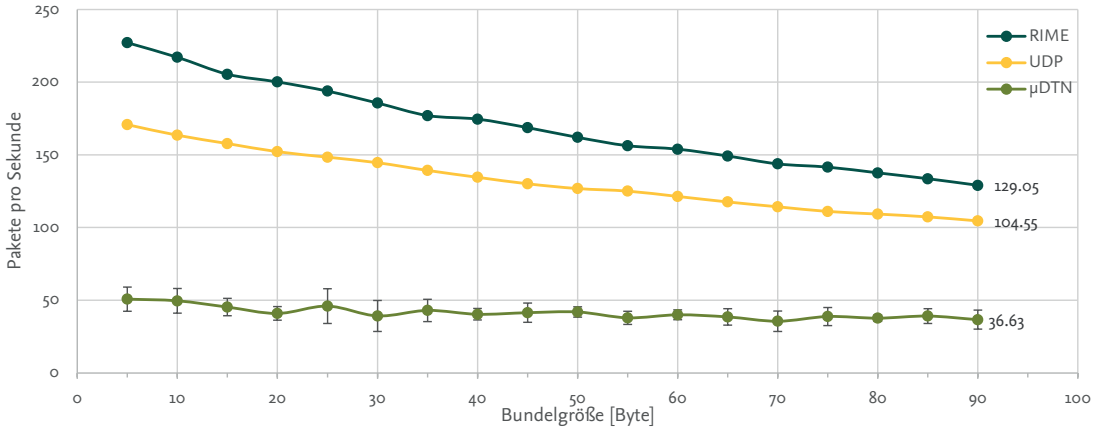


Abbildung 6.13: Die im Labor bestimmte Anzahl von Paketen bzw. Bundle pro Sekunde der Übertragungsprotokolle RIME, UDP und μDTN bei verschiedenen Paket- bzw. Bundlegrößen und einem Prozessortakt von 4 MHz.

Feldtests

Angepasst an den Anwendungsfall wurde μDTN auf INGA auch in realistischerer Umgebung evaluiert. Aus den Erfahrungen aus den Labortests wurde INGA dabei grundsätzlich mit 8 MHz betrieben. Die zu übertragenden Nutzdaten wurden dabei von INGAs Accelerometer generiert, welches mit konstanter Rate abgetastet wurde. Es wurden dabei Rohdatenpakete von jeweils 8 Byte Größe versendet, die die abgetasteten Werte der drei Achsen des Accelerometers (3·2 Byte) und einen 2 Byte Temperaturwert (aus dem Luftdrucksensor) enthielten.

Als Gegenstelle kam dabei die später in Kapitel 8 genauer erläuterte Basisastion (Multi-Services Home Platform) zum Einsatz. Als Empfänger wurde hier ein weiterer INGA-Sensorknoten verwendet, welcher über eine serielle USB-Verbindung mit der MSHP verbunden ist. Die empfangenen Daten wurden also vom Empfängerknoten empfangen, analysiert und seriell an die MSHP übermittelt.

Um den Anwendungsfall im Vordergrund zu halten, wurden in diesem Fall die Bundle mit bestimmten aber konstanten Datenraten erzeugt, während im Labortest der maximale Durchsatz bestimmt wurde, indem maximal große Bundle mit maximaler Geschwindigkeit erzeugt und versendet wurden. Dieses Vorgehen basiert auf der Annahme, dass die Sensordaten auch mit konstanter Datenrate erzeugt werden und die generierten Bundle daraus hervorgehen.

Um dennoch den erzielbaren Datenraten nahezukommen, wurden sowohl die Größe der Nutzlast als auch die Anzahl der pro Sekunde versendeten Bundle variiert. Die in Abbildung 6.14 dargestellten Datenraten sehen deshalb sehr linear aus, da lediglich die Kombinationen aus Nutzlastgröße und Bundle pro Sekunde erfasst sind, die problemlos und verlustfrei versendet und empfangen werden konnten.

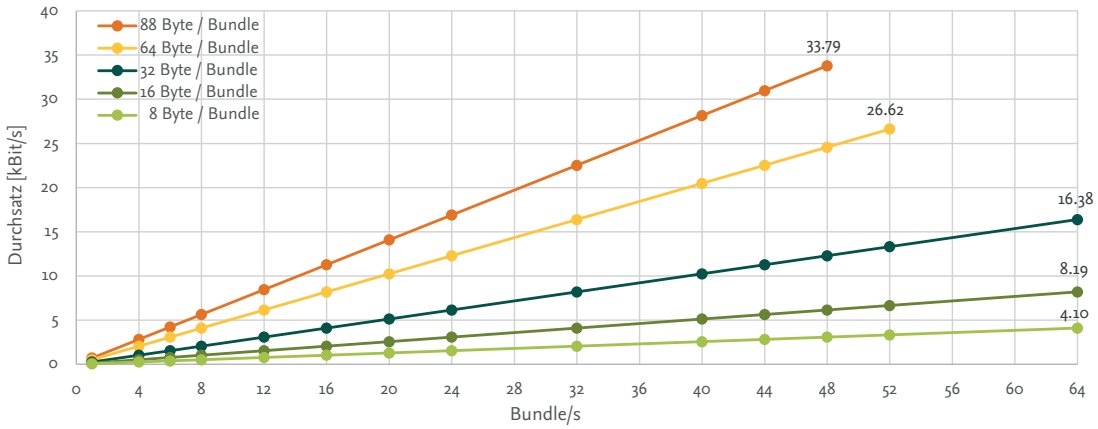


Abbildung 6.14: μDTN-Durchsatz bei verschiedenen Bundle-Größen und –Erzeugungsraten.

Der maximale Durchsatz liegt im betrachteten Fall also bei 33.79 kBit/s – bei einer Bundle-Nutzlast von 88 Byte und einer Erzeugungsrate von 48 Bundle pro Sekunde, was einer Abtastung der drei Achsen des Accelerometers mit 528 Hz entsprechen würde. Da nur durch 8 Byte teilbare Bundle erzeugt wurden und somit nicht die maximale Anzahl an Nutzdaten transportiert wurde, ist der Durchsatz also geringer als bei den Labortests. Das erklärt jedoch nicht einen solch großen Unterschied zwischen D_{Feld} =44.57 kBit/s und D_{Labor} =33.79 kBit/s.

Um dem auf den Grund zu gehen, wurde zum Vergleich auch der UDP-Durchsatz im betrachteten System analysiert und in Abbildung 6.15 dargestellt. Wie in Abbildung 6.9 zu sehen, hat UDP einen geringeren Overhead als μDTN und sollte folglich in der Lage sein, größere Datenmengen zu transportieren. Allerdings ist die UDP-Übertragung auch fehlerbehaftet, das heißt, dass Daten unwiederbringlich verloren gehen können. In Abbildung 6.15 ist im Bereich zwischen 70 und 75 Pakete/s (a) ein Einknicken aller Kurven zu erkennen: Ab dieser Anzahl von Paketen pro Sekunde ist der Empfänger nicht mehr in der Lage, alle Pakete auch zu verarbeiten und verwirft die Pakete entsprechend. Je nach Paketgröße ist der Sender in der Lage, mehr Pakete zu senden, als der Empfänger verarbeiten kann. In grau hinterlegten diagonalen Balken (b) in Abbildung 6.15 sind die maximal erzeugbaren Pakete pro Zeiteinheit zu erkennen – bei kleineren Nutzlastgrößen ist der Sender in der Lage, deutlich mehr Pakete zu erzeugen als bei größeren Nutzlasten. So können bei 8 Byte Payload bis zu 113 Pakete/s erzeugt werden – bei 64 Byte jedoch nur 85 Pakete/s. Auf den Nutzdatendurchsatz wirkt sich das jedoch nicht aus, lediglich die Paketverlustrate steigt, was dann auch in Abbildung 6.16 aufgetragen ist.

Dass der UDP-Durchsatz nicht dem in den Evaluationen in Kapitel 5 entspricht, ist wiederum der Tatsache geschuldet, dass auch hier das Gesamtsystem (siehe Abbildung 6.17) gemessen wurde und nicht der Durchsatz zwischen zwei ansonsten unbelasteten Knoten. Dasselbe wirkt sich auch auf den μDTN-Verkehr

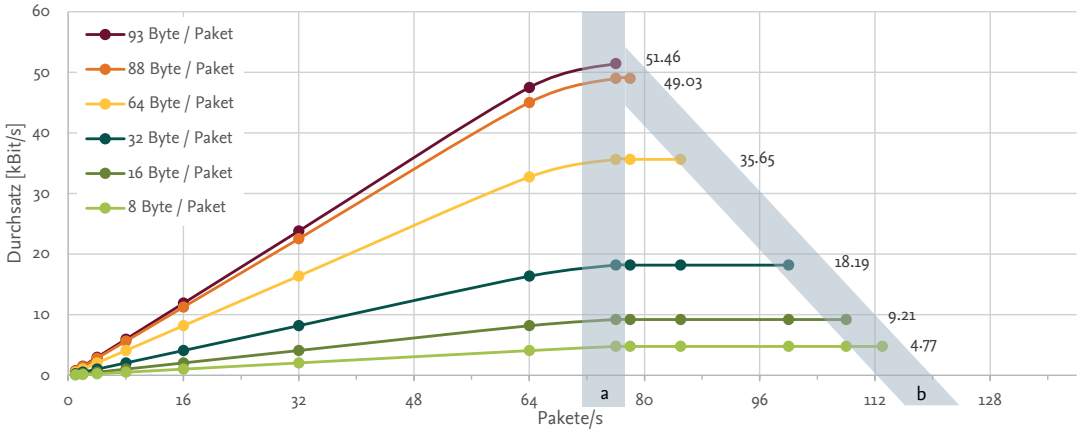


Abbildung 6.15: Erreichbarer UDP-Durchsatz.

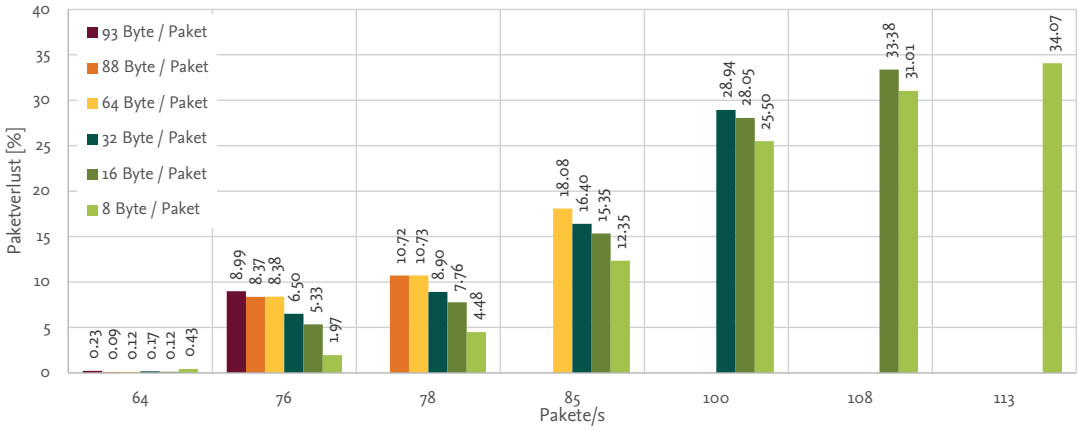


Abbildung 6.16: Paketverlust bei UDP-Verkehr bei bestimmten Erzeugungsdatenraten.

aus: Im Unterschied zu den Laborversuchen im vorherigen Abschnitt wird mit den Feldtests das gesamte System evaluiert. Der hinterlegte Kasten in Abbildung 6.17 entspricht dabei den unter Laborbedingungen durchgeführten Messungen des vorherigen Abschnitts.

In Abbildung 6.14 ist also die gemessene erreichbare Nutzdatenrate (Goodput) bei verschiedenen Nutzlastgrößen über die Anzahl der erzeugten Bundle pro Sekunde aufgetragen. Im untersuchten System traten bei einer großen Bundle-Rate und hoher Nutzlast am Empfänger vermehrt Paketverluste auf. Das rührt von der ebenfalls in Abbildung 6.17 dargestellten seriellen Verbindung zwischen Empfängerknoten und Senke her. Hier werden die empfangenen „ausgepackten“ und formatierten Daten über eine UART-Verbindung mit ressourcenintensiven *printf*-Befehlen gesendet. Dadurch wird der Empfängerknoten überlastet und das System wird instabil. Diesen Flaschenhals könnte man mit einem USB-Transceiver umgehen, der direkt die empfangenen Pakete an den Rechner weiterleitet. In diesem Fall sollten dann auch höhere Datenraten möglich sein. Bei den in Abbildung 6.14 aufgetragenen Datenraten kommt es jedoch zu keinen Paketverlusten.

In Abbildung 6.18 ist die Evaluation der grundsätzlichen Funktionalität von μDTN zu erkennen. Der Received Signal Strength Indication (RSSI)-Wert wird von INGAs Funktransceiver bereitgestellt und ist ein Indikator für die Empfangsfeldstärke. Der Link Quality Indicator (LQI)-Wert wird ebenfalls vom Transceiver berechnet, wobei hier auch die detektierten Übertragungsfehler mit einfließen. Wenn beide Werte fallen, ist

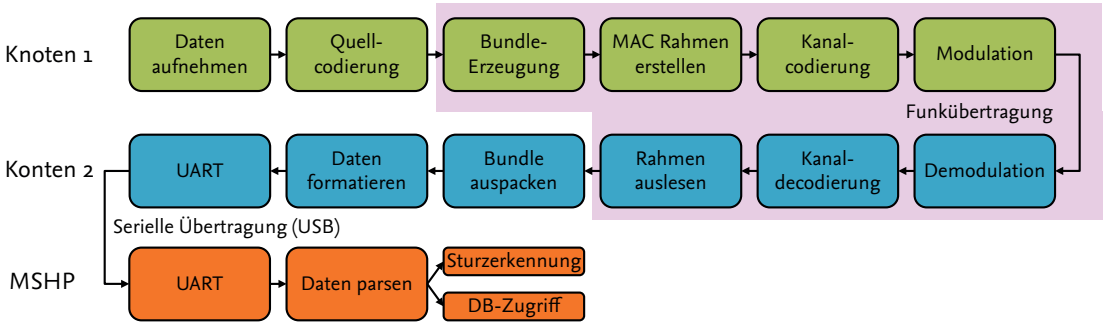


Abbildung 6.17: Das evaluierte Gesamtsystem vom Sensor bis zur Datensenke; der hinterlegte Kasten entspricht den μDTN-Messungen unter Laborbedingungen.

das ein Anzeichen dafür, dass der sendende Knoten die Kommunikationsreichweite verlässt; dieser Fall ist auf der linken Seite des Diagramms zu sehen. Sobald die Kommunikation abbricht, steigt der Pufferfüllstand und Bundle werden lokal zwischengespeichert, anstatt zugestellt zu werden; der mittlere Bereich des Diagramms zeigt dieses Verhalten. Mit erneutem Eintreten des mobilen Knotens in Kommunikationsreichweite der MSHP, steigen LQI und RSSI wieder an. Außerdem werden die zuvor zwischengespeicherten Daten übermittelt und so der Puffer in kurzer Zeit wieder geleert. Im konkreten Fall wurden alle 400 ms ein Bundle mit je 88 Byte Nutzdaten erzeugt; nach Wiederherstellung der Kommunikation wurden 17 Bundle innerhalb von 140 ms synchronisiert, was einem Synchronisationsdurchsatz von ~10.1 kBit/s entspricht. Da hier nur ein sehr kurzer Zeitraum betrachtet wurde, in den auch das Discovery fällt, wird in diesem Fall der maximale Durchsatz nicht erreicht.

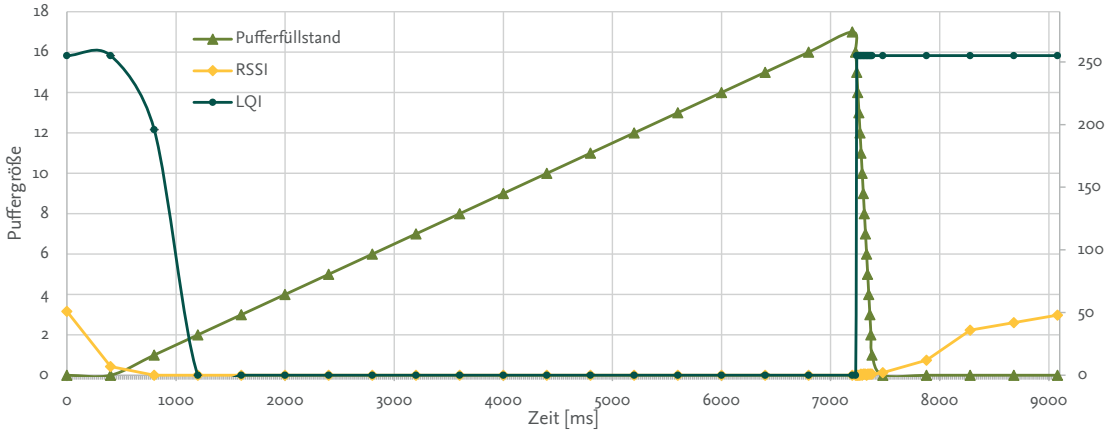


Abbildung 6.18: RSSI und LQI geben Auskunft über den Zustand des Funkkanals; der Pufferinhalt steigt mit zunehmender Unterbrechung des Funkkanals.

6.1.5 Zusammenfassung: Unterbrechungstolerante Netze

Nach einer Analyse der im medizinischen Umfeld typischerweise auftretenden Datenraten in Kapitel 4.2, wurde in diesem Abschnitt zunächst eine allgemeine Einschätzung der Leistungsfähigkeit von DTN-Protokollen gegeben. Es konnte gezeigt werden, dass DTN-Protokolle, wie z.B. das Bundle-Protokoll, in den betrachteten Anwendungsbereichen sinnvoll einsetzbar sind. Bei der Dimensionierung solcher Systeme ist dabei das Verhältnis zwischen erzeugter Datenrate und Durchsatz auf der Anwendungsschicht η die wichtigste Größe.

Dabei führen kleine Werte von η zu einer sehr kurzen Synchronisationsdauer, während bei η =0,5 die Synchronisierungszeit der Unterbrechungsdauer entspricht. Auch bei großen Werten ($0{,}5 < \eta < 1$) kann ein DTN sinnvoll sein, um kurzzeitige Unterbrechungen auf dem Funkkanal ohne Datenverlust zu überstehen. Daneben ist der zur Verfügung stehende Speicher von großer Bedeutung: Wenn $S_{fill} > S_{node}$ werden sollte, kommt es in jedem Fall zu einem Datenverlust, weswegen S_{node} entsprechend der maximal zu überbrückenden Zeit bzw. Datenrate großzügig gewählt werden sollte, was aber bei der Kapazität heutiger SD-Karten kein Problem darstellen sollte.

Bei Evaluationen im Labor und in Feldtests hat sich gezeigt, dass andere Protokolle (wie UDP und RIME) zwar einen quantitativ höheren Durchsatz erzielen konnten, jedoch nicht in der Lage waren, Verzögerungen und Unterbrechungen ohne Paketverlust zu überstehen. Die qualitative Anforderung an Unterbrechungstoleranz überwiegt dabei die rein quantitative Metrik des maximal erzielbaren Durchsatzes.

In der Evaluation des Gesamtsystems unter den Bedingungen eines Feldtests hat sich gezeigt, dass die zuvor unter Laborbedingungen ermittelte maximale Datenrate von ∼50 kBit/s im betrachteten System unter Realbedingungen so nicht erreicht werden kann. Jedoch liegt die erzielbare Datenrate mit gemessenen 33.79 kBit/s durchaus in derselben Größenordnung, weswegen weiterhin von einer grundsätzlichen Machbarkeit und Funktionalität ausgegangen werden kann. Da hierbei der Empfänger als limitierender Faktor identifiziert werden konnte, ist eine erste Verbesserung durch einen verbesserten Empfänger zu erreichen, welcher eine andere Art der Datenübertragung nutzt, als das Schreiben dieser Daten über eine serielle Verbindung. Ein Funkempfänger, der die empfangenen Daten direkt an ein Hostsystem weiterleiten kann, ohne sie vorher zu verarbeiten und zu analysieren, könnte so durch die Auslagerung der ressourcenintensiven Berechnungen zu einer Erhöhung des Durchsatzes beitragen.

Sollte der erzielbare Durchsatz auch dann nicht ausreichen, ist eine sinnvolle Reduktion der zu übertragenden Daten vorzusehen, was wiederum im übernächsten Abschnitt beschrieben wird. Zunächst wird im nächsten Abschnitt jedoch auf die Sicherheit und Verschlüsselung der zu übertragenden Daten eingegangen.

6.2 Sicherheit und Verschlüsselung

Bei Daten, die am menschlichen Körper aufgenommen werden, handelt es sich in der Regel um persönliche Daten, die einer Person zugeordnet werden können. Die mit einem BAN aufgenommen Vitalparameter oder Bewegungsdaten sind also inhärent privat. Diese Privatsphäre zu wahren ist nicht nur eine der Anforderungen zur Unterstützung der Szenarien (siehe Abschnitt 2.4), sondern entspricht auch geltendem Recht. Es ist also essentiell, die zu übertragenden Daten zu schützen, was wiederum eine ausreichend starke Verschlüsselung nahelegt.

Was aber ist eine ausreichend gute Verschlüsselung? Die fortlaufende Geschichte der Kryptologie hat sehr früh angefangen:

> „...wenn etwas Geheimes zu überbringen war, schrieb er in Zeichen, das heißt, er ordnete die Buchstaben so, dass kein Wort gelesen werden konnte: Um diese zu lesen, tauscht man den vierten Buchstaben, also D für A aus und ebenso mit den restlichen.“ [1]

Der hier beschriebene *Cäsar-Chiffre*, welcher von Julius Cäsar für seine *private* Korrespondenz benutzt wurde, beruht eher auf dem Prinzip *Security by Obscurity*, also *Sicherheit durch Unklarheit*. Es muss nicht extra erwähnt werden, das dieser Ansatz einfach zu knacken war – zu Cäsars Lebzeiten und natürlich ohne den Einsatz von Computern.

Die Geschichte der Verschlüsselung zeigt auch, dass fast jeder jemals zur Verschlüsselung eingesetzte Algorithmus mit der Zeit überwunden werden konnte, also die Verfahren auch immer nur für eine bestimmte

[1] C. Suetoni Tranquilli opera. Vol. 1. De vita Caesarum libri VIII. Teubner, Leipzig 1907. Editio minor 1908.

Zeit als sicher anzusehen sind. In der Vergangenheit konnten die meisten der eingesetzten Algorithmen, welche selbstverständlich als sicher und unüberwindbar gepriesen wurden, eben irgendwann doch geknackt werden – sei es wegen eines fehlerhaften Designs oder weil in der Zwischenzeit die Rechenkapazitäten gestiegen sind und so ein erfolgreicher Angriff mit einfach nur genug Rechenleistung durchgeführt werden konnte.

Um dem entgegen zu wirken, ist es gängige Praxis, entweder die Schlüssellängen für vorhandene Algorithmen zu erhöhen, oder aber die Algorithmen selbst gegen sicherere auszutauschen.

Beim Entwurf von tragbaren und eventuell sogar medizinischen Geräten, die dann in einem BAN zum Einsatz kommen, ist es mitunter nicht möglich, diese Geräte während ihrer Einsatzzeit anzupassen bzw. upzudaten, sobald sich die kryptologischen Gegebenheiten geändert haben und Verschlüsselungsverfahren angreifbar geworden sind:

Sicherheit Um ein System mit Updates zu versorgen, benötigt es eine Verbindung zu einem anderen System, welches diese Updates bereitstellt. Eine (weitere) Schnittstelle erhöht dabei auch immer die Angriffsfläche, so kann der Kanal, der eigentlich zur Verbesserung der kryptographischen Fähigkeiten vorgesehen wurde, in einem Angriffsszenario auch zum genauen Gegenteil genutzt werden.

Verteilung Geräte in einem BAN sind nicht ständig mit einer Infrastruktur verbunden und können auch über eine große geographische Fläche verteilt sein. Manche dieser Geräte werden aktiv genutzt, andere lagern möglicherweise ungenutzt über einen längeren Zeitraum. So ist es nur sehr schwer sicherzustellen, dass alle Geräte dieselbe Version der aktualisierten Kryptoroutinen erhalten haben. Unterschiedliche Versionen der Sicherungssysteme, die zur selben Zeit in einem Netz eingesetzt werden, können zu einem Ausfall des Gesamtsystems führen.

Leistungsfähigkeit Kleine eingebettete Systeme, wie Sensorknoten, haben nur sehr limitierte Leistungsreserven, da diese Geräte üblicherweise für den Einsatzzweck und in der Regel für eine lange Batterielaufzeit optimiert sind. Wenn verbesserte Kryptographiesysteme mehr Rechenkapazität beanspruchen, kann auch das zu einer Beeinträchtigung des Systems führen.

Hardwareunterstützung Aus Gründen der Ressourcenknappheit werden in Sensornetzen gerade Kryptographiefunktionen oft auf spezielle Hardwaremodule ausgelagert. So haben die meisten aktuellen Funktransceiver beispielsweise ein AES-Modul integriert, welches dem Mikrocontroller die Berechnung der AES-Verschlüsselung abnimmt. Einen solchen in Hardware implementierten Algorithmus auszutauschen, ist dann nur mit sehr großem (und meist physikalischem) Aufwand möglich.

Aber das sind nicht die einzigen Probleme, denen sich ein sicheres und zukunftsfähiges Kryptologiesystem für WSNs und BANs stellen muss. Im Folgenden werden daher zunächst einige grundlegende Verschlüsselungsmethoden erläutert. Anschließend wird die Implementierung des Vertreters mit der größten praktischen Relevanz auf dem INGA-Sensorknoten beschrieben und evaluiert. Im Anschluss daran werden dann Systeme zur Verschlüsselung für Sensornetze auf unterschiedlichen Schichten des TCP/IP-Referenzmodells diskutiert. Am Ende dieses Abschnitts wird ein Verschlüsselungssystem auf der Basis von One-Time-Pads für den betrachteten Anwendungsfall konzipiert, implementiert und evaluiert, welches sowohl die Verschlüsselung als auch den Schlüsselaustausch umfasst.

6.2.1 Verschlüsselungsmethoden

Die größte Herausforderung für Verschlüsselung und Authentifizierung in BANs und WSNs sind die limitierten Energie-, Rechen- und Kommunikationskapazitäten der eingesetzten Systeme [114]. Mit kleinen 8-Bit-Mikrocontrollern, die wenig Speicher besitzen und nur mit kleinen Taktraten laufen, wie sie bei INGA zum Einsatz kommen, können Verfahren, die aus der PC-Welt kommen, nicht ohne weiteres adaptiert werden.

Grundsätzlich kommen je nach Anwendungsbereich zwei unterschiedliche Verfahren zum Einsatz: *symmetrische Verschlüsselung* und *asymmetrische Verschlüsselung*. Bei symmetrischen Verfahren besitzen alle Kommunikationspartner denselben Schlüssel, der zum Verschlüsseln und zum Entschlüsseln der Nachrichten gleichermaßen verwendet werden kann. Symmetrische Verfahren sind einfacher zu berechnen und werden für größere Datenmengen eingesetzt.

Bei asymmetrischen Verfahren gibt es in der Regel ein Schlüsselpaar, welches aus dem privaten (*private*) und einem öffentlichen (*public*) Schlüssel besteht. Daten, die mit dem einen Schlüssel verschlüsselt wurden, können nur mit dem anderen Schlüssel entschlüsselt werden. Wenn also eine Nachricht von A zu B versendet werden soll, verschlüsselt A seine Nachricht mit dem öffentlichen Schlüssel von B; B kann diese Nachricht dann mit seinem geheimen privaten Schlüssel entschlüsseln. B kann so aber noch nicht sichergehen, dass die Nachricht tatsächlich von A kommt, da B's öffentlicher Schlüssel ja auch von jedem anderen benutzt werden kann. Eine Methode, die Authentifizierung mit asymmetrischen Verfahren sicherzustellen, wäre, das Vorhalten von einem Schlüsselpaar pro Kommunikationspartner, wobei dann der „öffentliche" Schlüssel jeweils auch geheim gehalten werden müsste. Es ist leicht erkennbar, dass ein solches System nicht sonderlich gut skaliert, da mit jedem weiteren Kommunikationspartner die Anzahl der benötigten Schlüssel steigt.

Eine Variante zur Authentifizierung, die mit weniger Schlüsseln auskommt, ist, dass A die Nachricht an B mit A's privatem Schlüssel verschlüsselt; B kann die Nachricht dann mit A's öffentlichem Schlüssel entschlüsseln. Auf diese Weise ist A als Absender durch B eindeutig zu identifizieren, aber jeder der A's öffentlichen Schlüssel kennt, ist in der Lage die Nachricht zu dechiffrieren. Insofern eignet sich das Verschlüsseln mit dem eigenen privaten Schlüssel auch nicht zur Wahrung von Vertraulichkeit, sondern nur zur Authentifikation; man spricht in diesem Fall vom *Signieren* der Nachricht, welches die Authentizität und die Verbindlichkeit der von A gesendeten Nachricht gewährleistet.

Um sowohl eine Authentifizierung als auch eine Verschlüsselung zu realisieren, werden in der Regel beide Verfahren miteinander kombiniert, d.h. A signiert die Nachricht zunächst mit seinem privaten Schlüssel und verschlüsselt sie anschließend mit B's öffentlichem Schlüssel. B kann die empfangene Nachricht nun mit seinem privatem Schlüssel entschlüsseln und mit A's öffentlichen Schlüssel verifizieren, dass die Nachricht auch von A kommt [115]. Die Integrität der Nachricht ist insofern gewährleistet, als dass jede Änderung der verschlüsselten Nachricht mit einer sehr großen Wahrscheinlichkeit zu einem komplett veränderten (und sinnlosen) Klartext führen würde; so ließe sich nur mit Kenntnis der entsprechenden Schlüssel die Nachricht durch Dritte auf dem Übertragungsweg verändern.

Mit einem solchen Verfahren wären die Anforderungen nach *Vertraulichkeit*, *Integrität*, *Authentizität* und *Verbindlichkeit* aus Abschnitt 2.4 erfüllt.

Solch asymmetrische Verfahren basieren allerdings auf komplexen mathematischen Operationen und sind deshalb nur sehr viel langsamer als symmetrische Verfahren zu berechnen; sie werden deshalb nur zur Verschlüsselung kleiner Datenmengen eingesetzt.

In der Praxis führt das dazu, dass in hybriden Verfahren die Vorteile der beiden Verfahren kombiniert werden. Mittels asymmetrischer Verschlüsselung werden Signaturen ausgetauscht und so die Authentizität der Kommunikationspartner sichergestellt. Außerdem wird in diesem Schritt ein gemeinsamer symmetrischer Schlüssel festgelegt und (bei vielen Verfahren) übertragen, mit dem dann der Austausch größerer Datenmengen symmetrisch verschlüsselt wird.

Asymmetrische Verschlüsselung in WSNs

Der am häufigsten eingesetzte Algorithmus zur asymmetrischen Verschlüsselung ist RSA [116], benannt nach seinen Entwicklern Ronald L. Rivest, Adi Shamir und Leonard M. Adleman. RSA basiert auf sehr großen Primzahlen und Einwegfunktionen, bei denen die eine Richtung leicht und die andere sehr schwierig zu berechnen ist: Eine Zahl mit einer Primzahl zu multiplizieren ist relativ einfach, während man annimmt,

dass die Primfaktorzerlegung ein weit größeres Problem ist. Wobei bei ressourcenschwachen Sensorknoten auch schon die „leichte“ Berechnung zu einer signifikanten Beeinträchtigung führen kann. Die Autoren von [117] beschreiben den Aufwand, der für die RSA-Berechnungen getätigt werden muss, als „zu viel“ für Sensorknoten. In durchgeführten Messungen haben sie für einen einfachen RSA-2048 Schlüsselaustausch mehr als 160 s gemessen.

In [118] werden RSA-Hardware- mit RSA-Software-Implementierungen verglichen. Es ist nicht verwunderlich, dass Hardware-Implementierungen ihre Software-Pendants in der Leistungsfähigkeit weit übertreffen. Jedoch dauert im dort betrachteten Szenario ein einfacher Schlüsselaustausch, durch mit speziellen Cryptochips ausgestatteten Sensorknoten, noch immer länger als eine Sekunde. Des Weiteren ist es erwähnenswert, dass die dort getätigten Untersuchungen auf Simulationen basieren, da solche RSA-hardwarebeschleunigten Sensorknoten einfach (bisher) noch nicht existieren.

Eine relativ neue Methode zur asymmetrischen Verschlüsselung basiert auf elliptischen Kurven über endlichen Feldern: Elliptic Curve Cryptography (ECC). Wie auch RSA basiert ECC auf einem nur sehr schwer lösbaren mathematischen Problem. Während es aber für RSA sub-exponentielle Lösungen gibt, ist die Lösung des ECC zugrundeliegenden „Problem des diskreten Logarithmus in elliptischen Kurven“ (ECDLP) vollständig exponentiell. Das führt dazu, dass sich bei ECC mit kleineren Schlüssellängen eine ähnlich hohe Verschlüsselungsstärke erreichen lässt, wie bei RSA mit deutlich längeren Schlüsseln. Ein 160-Bit-langer ECC-Schlüssel entspricht der Stärke eines 1024-Bit-RSA-Schlüssels, ein 224-Bit-ECC-Schlüssel einem 2048-Bit-RSA-Schlüssel. In [119] und [120] wird gezeigt, dass ECC-Algorithmen auf Sensorknoten etwas weniger Speicher belegen und dabei 4 bis 5 Mal schneller sind, als RSA-Implementierungen.

Symmetrische Verschlüsselung in WSNs

In [121] werden einige symmetrische Blockchiffren analysiert und mit Hinblick auf ihre Eignung für den Einsatz auf leistungsschwachen Sensorknoten evaluiert. Es gibt viele verschiedene symmetrische Verschlüsselungsmethoden und manche sind für spezielle Einsatzzwecke auch anderen vorzuziehen, jedoch wurde im Jahr 2001 der Rijndael-Algorithmus [122] vom amerikanischen National Institute of Standards and Technology (NIST) als „der“ *Advanced Encryption Standard (AES)* festgelegt. Ein Design-Ziel von AES war die Möglichkeit, auch auf schwächeren Prozessoren lauffähig zu sein. Erreicht wird das durch die Verwendung eines Substitutions-Permutations-Netzwerks (SPNs), welches aus einer Anzahl von mehreren Runden gleichem Aufbaus beruht, weswegen sich AES auch besonders für die Implementierung auf Mikrocontrollern anbietet. Der Standard definiert eine Schlüssellänge von 128, 192 oder 256 Bit und eine feste Blocklänge von 128 Bit

Nichtsdestotrotz basierten die ersten Implementierungen für Sensornetze nicht auf AES, sondern nutzten beispielsweise den Skipjack-Algorithmus, welcher auch sehr effizient zu implementieren ist, jedoch durch eine Vielzahl von Angriffen verwundbar ist [123]. TinySec [124] – die erste Implementierung für Verschlüsselung und Authentifizierung auf der Sicherungsschicht in TinyOS – benutzt den Skipjack-Algorithmus. In einer später zur Veröffentlichung hinzugefügten Fußnote räumen die Autoren allerdings ein, dass AES wohl eine ähnliche Performance auf der verwendeten Hardware hätte und ein „perfekt geeigneter Ersatz“ für Skipjack wäre.

Auch TinySecs Nachfolger MiniSec [125] und eine weitere Implementierung namens TinyKey [126] setzen auf den Skipjack-Algorithmus. Mit ContikiSec [127], einer Implementierung für die Sicherungsschicht für das Betriebssystem Contiki, wurde dann auch AES in Sensornetzen erfolgreich eingesetzt.

AES ist eine Blockchiffre und lässt sich als solcher in verschiedenen Modi betreiben. Die unterschiedlichen Modi haben auch einen Einfluss auf den erzielbaren Grad an Sicherheit.

Electronic Code Book (ECB)-Modus Der ECB-Modus ist der einfachste und (mit Abstand) unsicherste Modus von AES. In Abbildung 6.19 ist die prinzipielle Funktionsweise der Verschlüsselung dargestellt. Bei immer gleichen Eingaben und gleichem Schlüssel führt der ECB-Modus auch zu immer gleichen

Ausgaben. Da die Auftrittshäufigkeit von Blöcken im unverschlüsselten Text nur ausreichend verwischt werden, bietet er eine große Angriffsmöglichkeit für statistische Analysen. Der ECB-Modus wird von allen bekannten Hard- und Softwareimplementierungen unterstützt, allerdings sollte er nicht benutzt werden, wenn mehr als ein Block zu verschlüsseln ist [128].

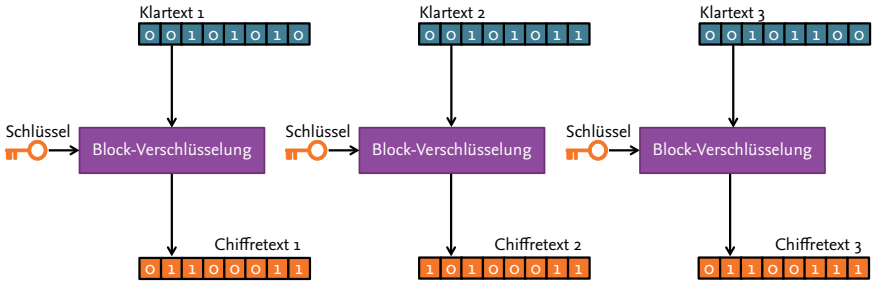


Abbildung 6.19: Der Electronic Code Book Mode (ECB) bei AES.

Cipher Block Chaining (CBC)-Modus Dieser deutlich sicherere Modus ist in Abbildung 6.20 dargestellt. Hier wird eine Kontravalenz (*XOR*) zwischen der Ausgabe der vorangegangenen Verschlüsselung und dem aktuell zu verschlüsselnden Klartext gebildet, wobei die erste Verschlüsselung mit einem beliebigen Vektor initialisiert wird. Auf diese Weise wird das Verschlüsselungsergebnis randomisiert, Klartextmuster zerstört und gleiche Klartextblöcke führen nun zu unterschiedlichen verschlüsselten Nachrichten. Eine Entschlüsselung funktioniert genau umgekehrt – hierbei wird dann eine Kontravalenz zwischen der verschlüsselte Nachricht der vorangegangen Übertragung mit der Ausgabe der aktuellen Entschlüsselung gebildet, um den korrekten Klartext zu erhalten; das setzt jedoch das Vorhandensein der vorausgegangenen Übertragung voraus. Bei *einer* fehlerhaften oder unvollständigen Übertragung sind also in jedem Fall *zwei* Blöcke betroffen: der verlorene und der, der den verlorenen Block zur Entschlüsselung benötigt.

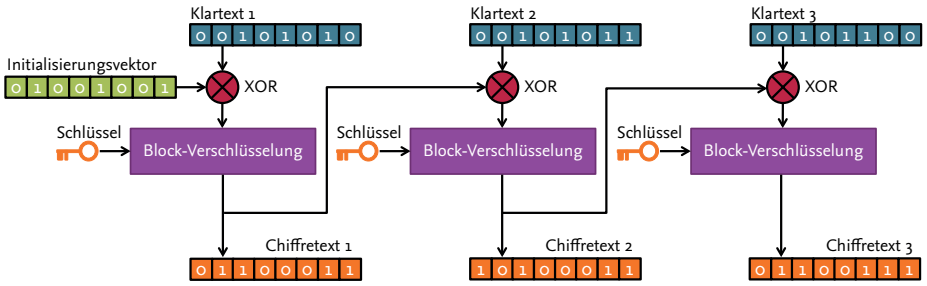


Abbildung 6.20: Der Cipher Block Chaining Mode (CBC) bei AES.

Weitere Modi Im Cipher Feedback (CFB)-Modus wird die Blockchiffre als Stromchiffre betrieben; das ist beispielsweise dann sinnvoll, wenn die zu verschlüsselnde Nachricht eine Länge aufweist, die kein Vielfaches der Blocklänge ist. Hierbei würde sich ein Übertragungsfehler ähnlich wie im CBC-Modus auf die Fähigkeit, die nächste Nachricht zu entschlüsseln, auswirken. Im Cipher Feedback (OFB)-Modus wird eine Synchronisation zwischen Sender und Empfänger benötigt. Wenn diese sichergestellt ist, hat dieser Modus den Vorteil, dass sich Fehler nicht fortpflanzen. Ähnliches gilt auch für den Counter (CTR)-Modus, bei dem der Initialisierungsvektor um einen Zähler erweitert wird und so sichergestellt wird, dass bei gleicher Klartexteingabe nicht derselbe verschlüsselte Text entsteht.

Trotz seiner Unsicherheit hat der ECB-Modus gegenüber allen andern Modi den Vorteil, dass keine weiteren Informationen vorhanden sein müssen, um auf die einzelnen Elemente zuzugreifen. So braucht eine im CBC-Modus verschlüsselte Nachricht ja zusätzlich den verschlüsselten vorherigen Block, um die aktuelle Entschlüsselung durchführen zu können. Bei einem stark gestörten Funkkanal, bei dem einige Nachrichten fehlerhaft oder gar nicht übermittelt werden, lassen sich also weniger Nachrichten wieder entschlüsseln, da die benötigten Informationen verloren wurden. Bei einer Paketverlustrate von 50 % ließe sich also im schlechtesten Fall, in dem nur genau jedes zweite Paket erfolgreich zugestellt wird, keinerlei Information rekonstruieren.

Beim OFB-Modus und beim CTR-Modus muss (ebenso wie beim CBC-Modus) die Synchronizität zwischen Sender und Empfänger sichergestellt sein, da sonst die Daten nicht korrekt entschlüsselt werden können. Die verschlüsselten Daten müssten also idealerweise in derselben Reihenfolge beim Empfänger ankommen; im Falle einer Übertragung über mehrere Zwischenstellen und in einem dynamischen Netzwerk ist das jedoch nicht zwangsweise gegeben, da unter Umständen verschiedene Pakete auf unterschiedlichen Pfaden dem Empfänger zugestellt werden.

Schlüsselverteilung und -austausch

Bevor jedoch überhaupt etwas verschlüsselt werden kann, bedarf es entsprechender Schlüssel auf den einzelnen Geräten. In der PC-Welt haben sich sogenannte *Public Key* Verfahren zum Schlüsselaustausch etabliert. Dabei wird zunächst über ein asymmetrisches Verfahren die Authentizität der Kommunikationspartner überprüft, um dann über einen ausgehandelten Sitzungsschlüssel die anfallenden Daten symmetrisch zu verschlüsseln. Hier kommen dann auch zentrale Zertifizierungsstellen (Certificate Authority (CA)) und TrustCenter zum Einsatz, welche die Authentizität von Kommunikationspartnern bestätigen können, die dem jeweils anderen Partner zuvor unbekannt sind. Für Ad-Hoc-Netze gibt es einige Ansätze zur Schlüsselverteilung, welche in [129] zusammengefasst sind.

In Sensornetzen, die ständig mit einer zentralen Instanz verbunden sind, könnte man auch auf CAs zurückgreifen, jedoch bleibt hierbei der Flaschenhals der benötigten asymmetrischen Verschlüsselung. Bei nur manchmal verbundenen Knoten und separierten Netzen ist der Einsatz einer zentralen CA jedoch sinnlos.

Die meisten Applikationen in WSNs setzen daher auf vorab verteilte Schlüssel. Dabei ist zu unterscheiden, ob lediglich eine Verschlüsselung gewünscht ist oder ob zusätzlich eine Authentifizierung angestrebt wird. Im ersten Fall würde dann wie bei ContikiSec ein Schlüssel für das gesamte Netzwerk vergeben. Das hat den Vorteil eines geringen Overheads, aber den Nachteil der einfachen Kompromittiererbarkeit. Wer Zugriff auf einen Knoten hat, kann den Schlüssel auslesen und ist dann in der Lage, den gesamten Verkehr im Netzwerk mitzuhören und selber (scheinbar valide) Daten zu senden. Ein einmal kompromittiertes Netzwerk ist – sofern die Kompromittierung überhaupt erkannt wird – dann auch nicht einfach zu bereinigen, da neue Schlüssel an alle beteiligten Knoten ausgegeben werden müssen.

Bei einer zusätzliche Authentifizierung würde jeder Knoten seinen eigenen Schlüssel haben und dieser müsste an alle anderen Knoten im Netzwerk verteilt werden. Bei großen Netzen mit n Knoten skaliert dieser Ansatz nicht gut, da $n - 1$ Schlüssel auf jedem Knoten gespeichert werden müssten. Eine Kompromittierung kann auch hier in der Regel erst an der Senke detektiert werden, wobei diese – entsprechende Mechanismen vorausgesetzt – dann auch an alle Knoten eine entsprechende Nachricht zum Ignorieren des kompromittierten Schlüssels versenden kann.

Um nicht alle Schlüssel aller Knoten in einem Netzwerk zu verteilen, gibt es auch die Idee der zufälligen Schlüsselverteilung (random key pre-distribution (RKS)), wie sie beispielsweise in [130] und [131] beschrieben wird. Hierbei gibt es jedoch immer nur eine gewisse Wahrscheinlichkeit, dass zwei Partner, die miteinander vertraulich kommunizieren wollen, dann dazu auch die benötigten Schlüssel haben.

Mit manchen Knoten als vertrauenswürdige Zwischensysteme, wurde dann mit PIKE [132] eine Lösung

umgesetzt, die es erlaubt eine vertrauenswürdige Kommunikation zwischen zwei benachbarten Partnern herzustellen. In [133] wird die Vertraulichkeit auf Basis von Gruppen hergestellt.

Weitere Verfahren, die einen tatsächlichen Schlüsselaustausch auf Basis von ECC realisieren, werden auch in Abschnitt 6.2.3 kurz vorgestellt.

6.2.2 Advanced Encryption Standard (AES) auf INGA

Unabhängig von der Verteilung der Schlüssel soll an dieser Stelle zunächst darauf eingegangen werden, wie kryptographische Verfahren auf INGA realisiert werden können. Bei den symmetrischen Verschlüsselungsverfahren ist also zunächst AES das Mittel der Wahl. Um zu bestimmen, mit welcher Performance AES auf INGA funktioniert und damit, ob es für den angedachten Anwendungsfall ausreichend ist, soll nun der Verschlüsselungsdurchsatz von AES bestimmt werden. Dazu werden mehrere Umsetzungen in Software miteinander und mit der Nutzung eines (auf INGA vorhandenen) Hardwaremoduls verglichen. Die Implementierungen umfassen dabei die folgenden Komponenten:

Einfaches Software AES – SW-AES-1 Die Referenzimplementierung des Rijndael-Algorithmus umfasst weniger als 500 Zeilen C-Code, was auch einer der Gründe war, weswegen er zum AES-Standard wurde. Anhand der Beschreibung des Algorithmus wurde AES für das Betriebssystem Contiki nachimplementiert, wobei die Blockgröße auf 128 Bit festgelegt wurde.

Erweitertes Software AES – SW-AES-2 In [134] wurden etliche Optimierungsmöglichkeiten für AES-Implementierungen vorgestellt. Um die einfache Implementierung zumindest etwas zu optimieren, wurde die direkte Implementierung um eine Wertetabelle erweitert, die allerdings weitere 1563 Byte an Programmspeicher belegt.

Referenzimplementierung in Assembler Um die eigenen C-Implementierungen für Contiki mit einer Art Referenz vergleichen zu können, wurde RijndaelFast[2] herangezogen – eine optimierte Assembler-Implementierung für die Atmel ATmega-Familie. Hierbei erfolgte keine Integration in Contiki, sodass auch spätere Tests, die Betriebssystemroutinen wie das Versenden von Daten nutzen, hiermit nicht durchgeführt werden konnten. Diese Implementierung soll in diesem Fall als theoretisch erreichbare Grenze für AES-Softwareimplementierungen gelten – wohl wissend, dass diese Werte bei zusätzlich vorhandenen Funktionalitäten nicht erreicht werden können.

Hardware AES auf INGA Der in INGA zum Einsatz kommende Funktransceiver AT86RF231 verfügt über ein integriertes AES-Sicherheitsmodul, welches AES-Kryptographie im ECB-Modus und im CBC-Modus ausführen kann. Dieses Modul funktioniert unabhängig von den zu versendenden Daten, d.h., es kann als einzelnes Feature verwendet werden und muss nicht auf die Datenrahmen angewendet werden. Für Contiki wurde ein entsprechender Treiber entwickelt, der dieses Hardwaremodul nutzen kann. Auf diese Weise können nun Hard- und Software-AES-Implementierungen auf INGA miteinander verglichen werden.

Evaluation

Die software- und hardwarebasierten AES-Implementierungen wurden auf INGA und mit Contiki (außer im Fall der Assembler-Referenzimplementierung) miteinander verglichen. Dabei wurde der Durchsatz der Verschlüsselung, der Datendurchsatz auf der Anwendungsschicht und der Speicherverbrauch (Footprint) gemessen.

[2] http://point-at-infinity.org/avraes/

Verschlüsselungsdurchsatz In Abbildung 6.21 ist der reine Durchsatz der einzelnen Implementierungen für die Ver- und Entschlüsselung aufgetragen, jeweils im ECB- und CBC-Modus. Die verbesserte Software-Version (SW-AES-2), welche von der zusätzlichen Wertetabelle profitiert, ist in allen Kategorien fast doppelt so schnell wie die direkte AES-SW-1 Umsetzung. Erkauft wird dieser Vorteil durch den, in Tabelle 6.2 zu erkennenden, zusätzlichen Speicherverbrauch. Die optimierte Assembler-Implementierung wiederum übertrifft die für Contiki in C implementierten Umsetzungen bei weitem. Dafür arbeitet diese Umsetzung nur alleine und nicht in Kombination mit anderen Funktionalitäten. Letztendlich ist aber besonders der große Unterschied der AES-Hardware-Implementierung zu den Softwareimplementierungen zu erkennen: Mit einem Verschlüsselungsdurchsatz im Bereich von > 1 MBit in jeder Konfiguration, lässt die Anbindung der Hardware-Verschlüsselung die Software-Umsetzungen weit hinter sich.

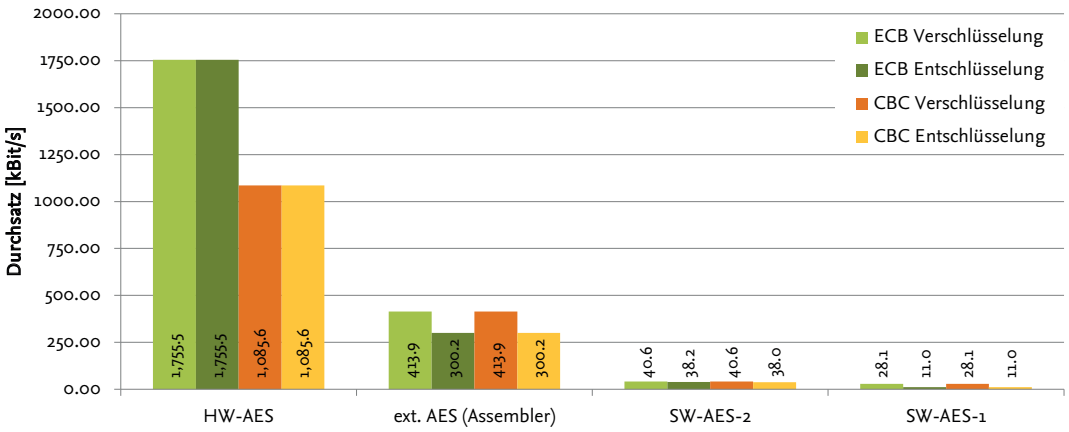


Abbildung 6.21: Verschlüsselungsdurchsatz der unterschiedlichen AES-Implementierungen.

Datenübertragung auf der Anwendungsschicht Um den Einfluss der Verschlüsselung auf die Gesamtperformance zu zeigen, wurde der Durchsatz von UDP-Datenverkehr mit unterschiedlichen Nutzlastgrößen bestimmt. Dazu wurden Daten mit den entsprechenden Algorithmen verschlüsselt und anschließend direkt versendet. In Abbildung 6.22 ist zu erkennen, dass auch die hardwarebeschleunigte Variante den erzielbaren Durchsatz des Gesamtsystems beeinflusst, allerdings längst nicht so stark, wie es die Algorithmen in Software tun. So ist auch mit angewandter Hardware-AES-Verschlüsselung noch ein UDP-Durchsatz von knapp 100 kBits möglich.

Speicherverbrauch Der Footprint in Programmspeicher (ROM) und Arbeitsspeicher (RAM) der einzelnen Implementierungen ist in Tabelle 6.2 zu sehen. Der ROM-Anteil wurde zur besseren Differenzierung in Daten (also Konstanten) und Funktionen aufgesplittet. Hier ist deutlich zu erkennen, dass SW-AES-2 seine bessere Performance durch mehr belegten Speicherplatz erkauft. Während beide Software-AES-Varianten nur 32 Byte an RAM verbrauchen, kommt die Hardware-Variante auf keinen zusätzlich benötigten Arbeitsspeicher. Auch der für die Hardwareanbindung benötigte Programmspeicher hält sich mit 518 Byte sehr in Grenzen.

Allgemein lässt sich aus diesen Daten ableiten, dass wenn eine Hardwareunterstützung für AES vorhanden ist, diese auch in jedem Fall zu nutzen ist. Speicherverbrauch, Geschwindigkeit und damit auch der Durchsatz auf der Funkverbindung sind jeweils deutlich günstiger, als jede optimierte Implementierung in Software.

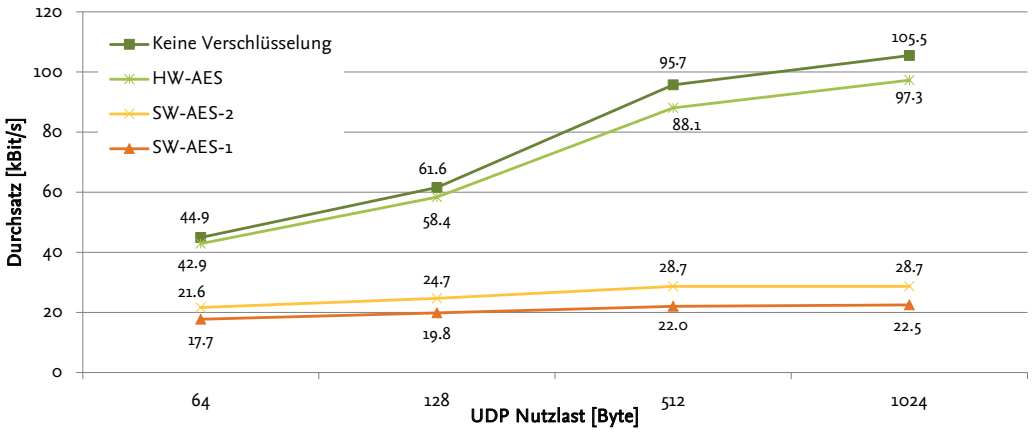


Abbildung 6.22: UDP-Durchsatz der unterschiedlichen Implementierungen der AES- Verschlüsselung.

Tabelle 6.2: Speicherbrauch der verschiedenen AES-Implementierungen

	RAM	ROM	
		Daten	Funktionen
SW-AES-1	32 Byte	522 Byte	2514 Byte
SW-AES-2	32 Byte	2058 Byte	2462 Byte
HW-AES	0 Byte	0 Byte	518 Byte

6.2.3 Verschlüsselung und Authentifizierung in drahtlosen Sensornetzen

Nachdem gezeigt wurde, welche Prinzipien und Algorithmen zum Einsatz kommen können und wie INGA in der Lage ist, diese zu unterstützen, soll nun ein Blick auf vorhandene Systeme geworfen werden, welche diese Algorithmen umsetzen.

Prinzipiell sind Verschlüsselung und Authentifizierung auf fast jeder Schicht des Netzwerkmodells anwendbar. Der IEEE-802.15.4-Standard sieht allerdings auf der Bitübertragungsschicht keine zusätzlichen Verfahren zur Erhöhung der Sicherheit vor. Eine Veränderung der Bitübertragungsschicht würde also dazu führen, dass der Standard verletzt würde und somit keine standardkonforme Kommunikation mehr möglich ist. Hinzu kommt, dass die Bitübertragungsschicht in der Regel in einem Mikrochip realisiert ist und sich so auch nicht ohne weiteres verändern lässt. In IEEE-802.15.4-Netzwerken Veränderungen auf der Bitübertragungsschicht vorzunehmen, würde also dazu führen, dass es keine Netzwerke nach IEEE 802.15.4 mehr sind. Auf der Sicherungsschicht hingegen sind Verschlüsselungs- und Authentifizierungsmethoden (teils optionaler) Teil des Standards.

Sicherungsschicht

In der Sicherungsschicht wird die Kommunikation zwischen zwei benachbarten Knoten definiert. Eine Ende-zu-Ende Sicherheit, die mehrere Hops umfasst, ist hier also technisch nicht einfach realisierbar. Für den betrachteten Anwendungsfall im aggregierten Szenario (siehe Kapitel 2.3.3) wäre eine alleinige Absicherung ausreichend, da dort zunächst nur ein Hop betrachtet wird. Spätere Erweiterungen, die dann auch mehrere Kommunikationspartner und eine Weiterleitung über mehrere Instanzen beinhalten, würden durch eine alleinige Absicherung der Sicherungsschicht jedoch ausgeschlossen werden.

Eine Authentifizierung auf der Sicherungsschicht würde dabei einen grundlegenden Schutz gegen unbekannte/ungewollte Netzwerkteilnehmer bieten, da unerwünschte Nachrichten nicht erst bei der Senke

sondern schon auf dem ersten Hop erkannt und dann nicht weitergeleitet würden.

CCM∗ im IEEE-802.15.4-Standard Die Sicherungsschicht selbst sieht mehrere Verfahren zur Verschlüsselung und Authentifizierung vor. CCM∗ ist eine im IEEE-802.15.4-Standard [93] vorgesehene Sicherungsmethode, die eine Verschlüsselung und Authentifizierung auf der Sicherungsschicht implementiert. Es werden vier verschiedene Sicherheits-Suiten definiert: *Keine Sicherheit*, *AES-CTR* zur Verschlüsselung mit AES im CTR-Mode, *AES-CBC-MAC* mit einem angehängten Message Authenticity Code (MAC) zur Sicherstellung von Authentizität und Integrität und *AES-CCM*, welches die beiden vorherigen Verfahren kombiniert. Prinzipiell können die Methoden zur Verschlüsselung in Hardware und zwar ohne Interaktion mit einem Betriebssystem ausgeführt werden. Dabei werden sie auf die im Puffer des Funktransceivers vorhandenen Daten angewendet, nachdem diese empfangen bzw. bevor diese gesendet werden.

Allerdings ergeben sich bei der Nutzung auch Einschränkungen, was die erlaubte Rahmengröße angeht. So werden für die AES-CTR-Verschlüsselung 5 Byte zusätzlich im Header des MAC-Rahmens benötigt. Je nach Größe des Message Authenticity Code, werden für die unterschiedlichen Versionen 4, 8 oder 16 zusätzliche Byte an die Nutzlast angehängt. Da die Rahmen insgesamt nicht länger als 127 Byte sein dürfen, ist das de-facto eine Verringerung der maximalen Rahmengröße. Mit AES-CCM gehen so – je nach Größe des Codes – 9 bis 21 Byte an Nutzlast zu Lasten der AES-CCM-Verschlüsselung verloren, was im konkreten Fall bei AES-CCM-128 zu einer Einbuße von $> 18\,\%$ führt. Das wiederum führt auch dazu, dass man CCM∗ nicht einfach „einschalten“ und benutzen kann, da die zuvor verwendeten Rahmenstrukturen unter Umständen anschließend zu groß sind.

TinySec TinySec [124] wurde schon zu Beginn des Kapitels als Sicherheitsarchitektur auf der Sicherungsschicht von TinyOS vorgestellt. TinySec besitzt dabei zwei Operationsmodi: *Authenticated Encryption* (TinySec-AE) und *Authentication only* (TinySec-Auth). Im ersten Fall wird dabei die Nutzlast mit Skipjack im CBC-Modus verschlüsselt und anschließend ein MAC über den Header und die verschlüsselte Nutzlast berechnet. Im zweiten Fall wird einfach ein MAC über das gesamte Paket berechnet. Da TinySec auch eine Optimierung der Protokolle beinhaltet und so beispielsweise das *Group*-Feld aus dem normalen TinyOS-Paket-Format entfernt, ist der Kommunikationsoverhead eher gering. Während TinySec-AE 5 Byte zusätzlich beansprucht, belegt TinySec-Auth lediglich ein weiteres Byte.

ContikiSec Auch ContikiSec [127] wurde zu Beginn des Abschnitts schon als Vertreter der AES-benutzenden Suiten erwähnt. ContikiSec definiert drei verschiedene Operationsmodi: Vertraulichkeit (ContikiSec-Enc), Authentizität (ContikiSec-Auth) und die Kombination (ContikiSec-AE). Während ContikiSec-Enc einen Overhead von 2 Bytes pro MAC-Frame verursacht, hängt ContikiSec-Auth einen 4 Byte MAC an die Nutzlast an. Da aber gleichzeitig auch die Prüfsumme entfernt wird, ergibt sich auch für ContikiSec-Auth nur ein Overhead von 2 Bytes. Konsequenterweise verursacht ContikiSec-AE als Kombination der beiden zuvor genannten einen Gesamtoverhead von 4 Byte. Eine Einschränkung ist allerdings, dass ContikiSec davon ausgeht, dass alle beteiligten Knoten denselben 128 Bit Schlüssel verwenden, welcher von vornherein auf den Knoten installiert sein muss.

Per Definition können Protokolle auf der Sicherungsschicht nur eine Hop-zu-Hop Sicherung gewährleisten und so wäre jeder Knoten im Netzwerk, der in der Lage ist, eine Nachricht erfolgreich zu empfangen und weiterzuleiten, auch in der Lage, den Inhalt zu lesen – solange die Nutzdaten nicht auch noch durch Protokolle darüber liegender Schichten gesichert sind.

Vermittlungsschicht

Die Vermittlungsschicht bietet eine Ende-zu-Ende-Verbindung, insofern ist hier auch eine Ende-zu-Ende-Verschlüsselung implementierbar.

Internet Protocol Security (IPsec) In [135] wurde eine komprimierende IPsec-Implementierung vorgestellt, welche auf 6LoWPAN aufsetzt. Hierbei wird AES zur Verschlüsselung und SHA1 zur Authentifizierung eingesetzt. Der IPsec-Standard sieht allerdings auch ein Internet Key Exchange (IKE)-Protokoll zum Schlüsselaustausch vor, welches hier nicht umgesetzt wurde. Das heißt, dass die Umsetzung wie auch schon ContikiSec auf zuvor verteilte Schlüssel setzt. In ihrer Evaluation geben die Autoren einen Kommunikationsoverhead von 30 Byte in der unkomprimierten und 24 Byte in der komprimierten Variante ihrer IPsec Implementierung an. Zur Beschleunigung der AES-Verschlüsselung werden auch eine Hardwarebeschleunigung in Betracht gezogen und deren Vorteile aufgezeigt.

Protokolle in der Vermittlungsschicht setzen andere Protokolle auf der Transportschicht voraus. So können über ein mit IPsec abgesichertes 6LoWPAN zwar UDP- und Transmission Control Protocol (TCP)-Verbindungen aufgebaut werden, andere Protokolle wie RIME oder μDTN können aber nicht davon profitieren, da sie nicht auf 6LoWPAN aufsetzen.

Transportschicht

Auch auf der Transportschicht wird eine Ende-zu-Ende-Verbindung aufgebaut und somit lässt sich auch hier eine Ende-zu-Ende-Verschlüsselung realisieren.

Sizzle Erstmals wurde mit Sizzle [119] die Implementierung eines HyperText Transfer Protocol Secure (HTTPS)-Stacks für drahtlose Sensornetze mittels Secure Sockets Layer (SSL)/Transport Layer Security (TLS) vorgestellt. Hierbei kommt auch mit ECC ein asymmetrisches Verschlüsselungsverfahren zum Schlüsselaustausch zum Einsatz, während die Verschlüsselung des Datenverkehrs über RC4 vorgenommen wird. In ihrer Evaluation konnten die Autoren zeigen, dass TelosB-Knoten einen Durchsatz von $\sim$34 kBit/s über eine normale HTTP-Verbindung erreichen konnten. Mit Sizzle konnten die Knoten dann $\sim$28 kBit/s über eine verschlüsselte HTTPS-Verbindung zwischen zwei Knoten austauschen. Das heißt, dass die ohnehin schon langsame HTTP-Übertragung noch einmal um $\sim$18 % verlangsamt wurde. Bei Mica2-Knoten, die einen ähnlichen Prozessor wie INGA verwenden, konnte über eine normale HTTP-Verbindung nur $\sim$6 kBit/s und über eine verschlüsselte HTTPS-Verbindung $\sim$5 kBit/s übermittelt werden.

μDTN: Bundle Security Protocol Als Cross-Layer-Ansatz ist μDTN zwar nicht nur ein Transportprotokoll, aber da es ein API zur Anwendungsschicht anbietet, soll es an dieser Stelle zumindest erwähnt werden: Da ja μDTN eine Bundle-Protokoll konforme Implementierung ist, liegt es auch nahe, eventuelle Sicherheitserweiterungen des Bundle-Protokolls in Betracht zu ziehen. In [136] wurden die Spezifikationen für ein *Bundle Security Protocol* vorgestellt, wie sie beispielsweise in Bundle-Protokoll-Implementierungen aus der PC-Welt wie IBR-DTN bereits umgesetzt sind. Es werden vier Arten von *Bundle Security Blöcken* definiert, die in einem Bundle enthalten sein können: der Bundle Authentication Block (BAB), der Payload Integrity Block (PIB), der Payload Confidentiality Block (PCB) und der Extension Security Block (ESB). Außerdem werden für jeden dieser Blöcke verbindliche zu unterstützende Verschlüsselungsmethoden vorgeschrieben. Bis auf den BAB, der für eine Authentifizierung zwischen 2 Knoten zuständig ist, werden jedoch für alle weiteren Blöcke zumindest Schlüsselaustauschalgorithmen auf RSA-Basis vorgeschriebenen, was – wie schon erwähnt – den Einsatz in Sensornetzen deutlich erschwert.

So kann insgesamt davon ausgegangen werden, dass eine protokollgerechte Umsetzung des *Bundle Security Protocol* auf leistungsschwachen Sensorknoten zwar nicht unmöglich, jedoch aber sehr ressourcenintensiv ist. Außerdem würde eine den Spezifikationen entsprechende Umsetzung mindestens 10 Byte Overhead pro Bundle bedeuten.

Aus Sicht der PC-Welt mag eine Absicherung der Transportschicht durchaus Sinn ergeben und auch wenn man mit einem PC gesichert auf Sensorknoten zugreifen möchte, kann der Einsatz von HTTPS sinnvoll sein.

Für viele Anwendungen im Bereich von Sensornetzen ist aber der Overhead, den eine HTTPS-Verbindung benötigt, einfach zu groß.

Anwendungsschicht

Während sich Anwendungen auf die Sicherungsmechanismen darunterliegender Protokolle verlassen können, können sie aber (zusätzlich) auch eigene Mechanismen definieren. Ein Nachteil einer individuellen Verschlüsselung auf Anwendungsebene ist jedoch, dass man diese wahrscheinlich für jede Anwendung neu implementieren muss.

Würde man jedoch nur die Nutzlast verschlüsseln, ginge das – je nach Methode – nicht zu Lasten des Durchsatzes, vorausgesetzt der Knoten ist in der Lage, die Verschlüsselung ohne großen Rechenoverhead durchzuführen. Ein weiterer Vorteil der Verschlüsselung auf Anwendungsebene wäre die Unabhängigkeit von darunterliegenden Kommunikationsprotokollen. Während HTTPS zwingend TCP und IP voraussetzt, können bei einer reinen Nutzlastverschlüsselung beliebige Protokolle der Transport-, Vermittlungs- und Sicherungsschicht verwendet werden.

In Abbildung 6.23 sind die unterschiedlichen Ansätze auf den einzelnen Schichten noch einmal zusammenfassend dargestellt. Im folgenden Abschnitt wird ein Entwurf und eine Implementierung für eine Verschlüsselung auf Anwendungsschicht vorgestellt.

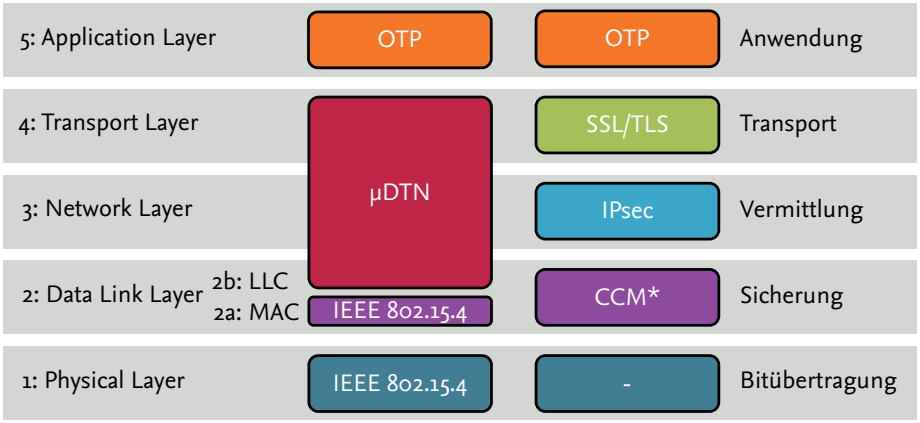


Abbildung 6.23: Unterschiedliche Ansätze zur Verschlüsselung in unterschiedlichen Schichten des TCP/IP-Referenzmodells (rechts) und die OTP-Implementierung (links)

6.2.4 One-Time-Pads

Die Idee von Einmalschlüsseln (One-Time-Pads) ist nicht neu; sie wurde zuerst von Gilbert Vernam (Bell Labs) und Joseph Oswald Mauborgne im Jahr 1918 beschrieben. Im Kalten Krieg wurde eine auf OTPs basierende Fernschreiberverbindung als sogenannter „Heißer Draht“ zwischen Moskau und Washington betrieben.

Die Funktion von OTPs ist so einfach wie überzeugend und ist in Abbildung 6.24 dargestellt. Eine Kontravalenz von Klartext und Schlüssel ergibt den chiffrierten Text – oder einfacher ausgedrückt: Der Klartext wird mit einem Schlüssel derselben Länge *XOR*ed. Es handelt sich also um eine symmetrische Verschlüsselung, die sehr einfach zu berechnen ist.

In den 1940er Jahren hat Claude Shannon den mathematischen Beweis erbracht, dass eine OTP-Verschlüsselung, wenn sie richtig angewendet wurde, nicht zu brechen ist [137]. Kryptosysteme, die auf OTPs basieren, sind *informationstheoretisch sicher* bzw. *kryptoanalytisch nicht zu brechen*. Selbst mit unlimitierter Rechenleistung gibt es keine Chance den Klartext zu berechnen, da einfach nicht genug Informationen für einen Angriff zur Verfügung stehen. Da OTPs (mindestens) genau so lang wie der zu verschlüsselnde Text sein müssen, kann

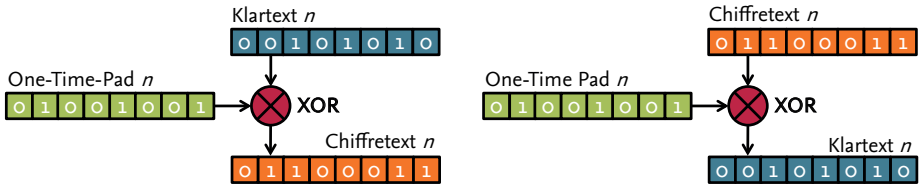


Abbildung 6.24: Grundsätzliche Funktionalität von One-Time-Pads: Bei der Verschlüsselung (links) ergibt sich der Chiffretext aus der Kontravalenz von Klartext und Schlüssel. Die Entschlüsselung (rechts) funktioniert analog.

ein verschlüsselter Text nur mit dem korrekten Schlüssel zum richtigen und sinnvollen Klartext entschlüsselt werden. Es existieren aber auch beliebig viele falsche Schlüssel, mit denen man ebenfalls einen (vermeintlich) sinnvollen Text zu erhalten scheint.

Für die korrekte Implementierung einer OTP-basierten Verschlüsselung müssen nur ein paar Regeln bezüglich des *Pads* (was ein Synonym für *Schlüssel* ist) eingehalten werden:

1. Das Pad muss mindestens so lang wie der zu verschlüsselnde Klartext sein.
2. Das Pad muss gleichverteilt zufällig erzeugt worden sein.
3. Das Pad muss geheim bleiben.
4. Ein und dasselbe Pad darf nur einmal verwendet werden.

Die erste und die letzte Regel machen das System in der Regel schwer zu handhaben, widersprechen sie doch den Zielen anderer Verschlüsselungsverfahren, dass der Schlüssel möglichst kurz zu sein hat, damit man ihn auch erinnern und im Zweifel von Hand eintragen kann. Im Folgenden wird daher gezeigt werden, wie man dieses Problem mit ausreichend zur Verfügung stehendem Speicher lösen kann.

Datenraten und Speicherbedarf in BAN-Anwendungen

In Abschnitt 4.2 wurden in Tabelle 4.1 die unterschiedlichen Datenraten, die von BAN-Sensorik erzeugt werden können, aufgezeigt. In Abschnitt 6.1 wurde bereits eine Einschätzung über die in einem BAN entstehenden und zu übermittelnden Datenmengen gegeben. In Tabelle 6.1 wurde beispielhaft für die Datenaufzeichnung eines Accelerometers mit $G_{gen} = 1500\,\text{Bit/s}$ gezeigt, wie lange eine Synchronisierung dauern würde, aber auch wie viel Speicher zur Zwischenspeicherung gebraucht würde.

Die im Anwendungsfall anvisierte Aufzeichnung von Accelerometerdaten mit $G_{gen} = 1500\,\text{Bit/s}$ würde demnach in einer Sekunde 187.5 Byte an Daten produzieren. In einer Stunde wären es 675.00 KByte und in einem Tag 16.20 MByte – ein Monat würde 493.07 MByte und ein Jahr 5.65 GByte an Speicherplatz benötigen. Für diese Datenmenge muss also dieselbe Menge an OTPs auf dem Sensorknoten vorgehalten werden, um die Daten entsprechend mit OTPs zu verschlüsseln.

Speicherkartenhersteller haben gerade SD-Karten mit einer Kapazität von 256 GByte auf den Markt gebracht; im kleineren microSD-Format sind mittlerweile Karten mit 128 GByte im Handel erhältlich. Mit solchen Speicherkapazitäten lassen sich also Zeiträume von über 20 (128 GByte) bis über 40 (256 GByte) Jahren abdecken – sofern die Geräte so lange halten.

Um noch ein weiteres Beispiel zu geben: Mit einem OTP-Speicher von 64 GByte lassen sich die Daten, die ein 12-Kanal-EKG erzeugt, welches mit 16 Bit Auflösung und 500 Hz Abtastfrequenz aufzeichnet, für mehr als zwei Monate mit OTPs verschlüsseln. Somit ist ersichtlich, dass sich nicht nur die relativ geringen Datenraten des Accelerometers sondern auch weit höhere Datenraten sinnvoll mit OTP verschlüsseln lassen.

Zusammenfassend lässt sich sagen, dass One-Time-Pad für alle Zeiten sicher sind, und dass die gute und kostengünstige Verfügbarkeit von großen Speicherkapazitäten einen Einsatz von OTPs für die Absicherung der Daten, die in BANs oder WSNs aufgezeichnet werden, unterstützt. OTPs können also genutzt werden, um eine verschlüsselte Übertagung von privaten Daten abzusichern, sofern auch die benötigten Schlüssel auch geeignet erzeugt und verteilt werden können. Deshalb soll nun ein System zur Erzeugung, Übermittlung und Nutzung von OTPs entworfen und implementiert werden, um es anschließend zu evaluieren.

Systementwurf

In Abbildung 6.25 sind die grundsätzlichen Funktionsweisen eines Systems zum Absichern von übermittelten Daten eines WBAN mittels OTPs aufgezeigt. Dieses System bietet im Gegensatz zur alleinigen (oben erwähnten) Vorinstallation von OTPs auf den Knoten zusätzlich die Möglichkeit, OTPs „aufzuladen“, weswegen die Grafik aus zwei Teilen besteht: Im linken Teil (A) der Abbildung ist die grundsätzliche Funktionalität der Verschlüsselung mit OTPs zu sehen. Ein Knoten in einem WBAN nimmt Daten auf, verschlüsselt diese mittels entsprechenden OTPs und sendet diese verschlüsselten Daten dann zur Basisstation, sofern diese in Funkreichweite (d) ist.

Im rechten Teil (B) der Grafik ist eine Erweiterung dargestellt. Wenn der Sensorknoten gerade nicht getragen wird – also beispielsweise nachts oder nach Ende der Studie – kann der OTP-Speicher wieder aufgefüllt werden. Dazu wird der Knoten per Universal Serial Bus (USB) an die Basisstation angeschlossen, die sowohl den Akku des Knotens laden als auch neue OTP auffüllen kann.

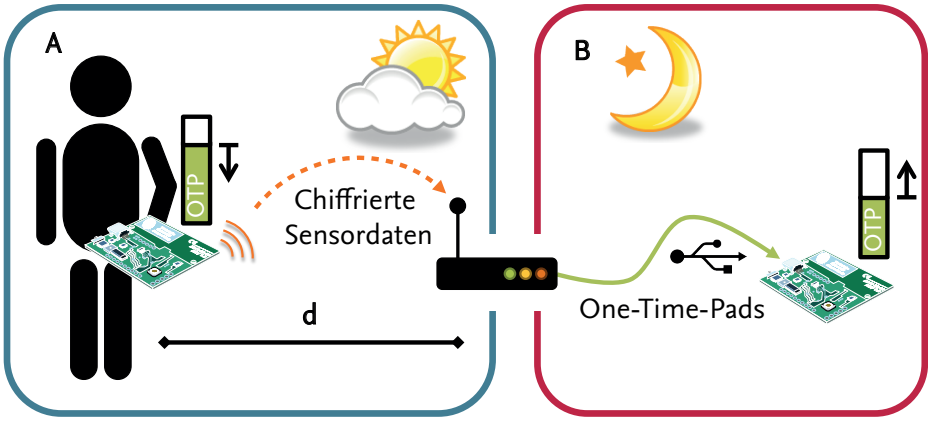


Abbildung 6.25: System zur Erzeugung, Übertragung und Nutzung von One-Time-Pads.

Zufallszahlengenerator In der Vergangenheit und auch während dieses Dokument geschrieben wird, hat es eine Vielzahl von erfolgreichen Angriffen auf Kryptosysteme gegeben. Viele dieser Angriffe gingen auf einen unzureichend implementierten Zufallszahlengenerator zurück [138]. In diesem Fall liefert der Generator nicht wirklich zufällige Ergebnisse, sondern nur ein kleineres Subset von bekannten Zahlen. So wird der Lösungsraum signifikant eingeschränkt und es wird Angreifern die Möglichkeit geboten, selbst vermeintlich sichere Verfahren zu attackieren. Die Verfügbarkeit von echten und gleichverteilten Zufallszahlen ist somit essentiell für nahezu jedes Kryptosystem.

Heutige Betriebssysteme bieten in der Regel einen Zufallszahlengenerator an, der „gut genug“ ist – auf jeden Fall solange, bis das Gegenteil bewiesen ist. Um sich nicht auf Zufallsgeneratoren verlassen zu müssen, die das Betriebssystem bereitstellt, können auch Zufallsgeneratoren in Hardware verwendet werden. In [139] wurden die empfangenen Bitfehler eines IEEE-802.15.4-Transceivers verwendet, um echte, durch einen zufälligen physikalischen Prozess determinierte Zufallswerte zu erhalten. Etwas in

dieser Art kann auch für die Basisstation implementiert werden, da hier zum Empfangen der Daten aus dem WBAN ein solches Funkinterface ebenfalls vorhanden ist.

Pad-Austausch Nachdem die Pads durch einen adäquaten Zufallsgenerator erzeugt wurden, wird eine Kopie auf der Basisstation gespeichert und eine weitere Kopie auf den Knoten transferiert. Um die OTPs von der Basisstation auf den Knoten transferieren zu können, muss ein sicherer Übertragungskanal vorhanden sein. Das mit OTPs verschlüsselte Übertragen von neuen OTPs würde nicht viel Sinn ergeben, da beide dieselbe Länge hätten und somit nichts gewonnen wäre. Das Mittel der Wahl ist in diesem Fall der Transfer über ein abgeschirmtes Kabel. Immer wenn der Knoten mit der Basisstation verbunden ist, können neue OTPs übertragen werden. Ein USB-Kabel zwischen Basisstation und Knoten kann also simultan den Akku laden und OTPs austauschen.

Eine weitere Methode zum Austausch bzw. zur Erneuerung von OTPs könnte das Austauschen der SD-Karten sein. Immer wenn der OTP-Speicher aufgebraucht ist, würde der Nutzer eine neue SD-Karte, die zuvor von der Basisstation mit „frischen " OTPs bestückt wurde, einsetzen und wäre anschließend wieder für einen gewissen Zeitraum in der Lage, verschlüsselt zu kommunizieren. In diesem Fall wäre auch ein PC-kompatibles Dateisystem für die SD-Karte (wie beispielsweise FAT) von Vorteil.

Speicherbenutzung Für die Nutzung von OTPs auf den drahtlosen Sensorknoten können sich drei verschiedene Szenarien ergeben, die in Abbildung 6.26 dargestellt sind. Im ersten Fall (a) werden alle aufgenommenen Daten direkt nach dem Aufnehmen verschlüsselt und dann umgehend an die Datensenke gesendet. Hierbei kann der komplette auf Knoten und/oder SD-Karte zu Verfügung stehende Speicher initial mit OTPs gefüllt werden. Die benutzen OTPs können dann entweder als bereits benutzt markiert oder gelöscht werden, sodass der Speicher anschließend wieder für andere Daten zur Verfügung steht.

Wenn die verschlüsselten Daten nicht direkt übertragen werden können, weil beispielsweise gerade keine Funkverbindung zur Verfügung steht, müssen die Daten auf dem Knoten zwischengespeichert werden. Im Fall (b) werden die Daten direkt und an Ort und Stelle („in place") verschlüsselt und der Anteil der verschlüsselten Daten wächst mit dem Abnehmen der zur Verfügung stehenden OTPs. Auch in diesem Fall kann der insgesamt zur Verfügung stehende Speicher von vornherein mit OTPs gefüllt werden.

In Szenarien, in denen nur Daten aufgenommen, aber nicht weiterverarbeitet werden, sind (a) oder (b) die wahrscheinlichsten Umsetzungen. Wenn aber eine Datenverarbeitung auch auf dem Knoten stattfinden soll, müssen dafür auch unverschlüsselte Daten zur Verfügung stehen.

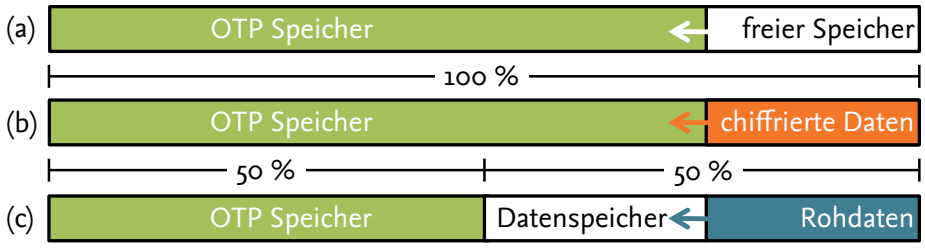


Abbildung 6.26: Unterschiedliche Speicherverwendung bei der Verwendung von OTP-Verschlüsselung.

Algorithmen zur Sturzerkennung oder Ganganalyse beispielsweise, die auf den Daten der vergangenen Zeit arbeiten, können nur dann funktionieren, wenn diese Daten auch im Klartext vorliegen. In der Regel umfasst dieser Zeitraum nicht mehr als 10 Minuten, aber im schlimmsten Fall müssen – je nach Anwendung – alle Daten auch zur Verarbeitung im Klartext auf dem Knoten vorhanden sein. In diesem

Fall, dass alle Daten auf dem Knoten in verarbeitbarer Form vorliegen müssen und außerdem keine Daten versendet werden können, kann nur die Hälfte des auf dem Knoten zur Verfügung stehenden Speichers auch für OTPs verwendet werden. Die andere Hälfte wird dann für das Zwischenspeichern von unverschlüsselten Rohdaten benötigt, was mit (c) verdeutlicht ist.

Aus Sicht der Vertraulichkeit sollte aber das Speichern unverschlüsselter Daten auf dem Knoten (soweit es machbar ist) vermieden werden. Trotz alledem bliebe auch in diesem Fall die Funkübertragung verschlüsselt und nur wenn das Gerät (physikalisch) in unbefugte Hände gerät, könnte es kompromittiert werden.

Einschränkungen Nichtsdestotrotz, sobald eines der beteiligten Geräte – sei es Sensorknoten oder Basisstation – kompromittiert wird und eine unbefugte Partei uneingeschränkten Zugang zur Hardware erlangt, kann die Datenübertragung nicht mehr als sicher angenommen werden – wie auch bei allen anderen kryptographischen Systemen. Bei Verlust eines mobilen Knotens müssen die Daten dieses Knotens verworfen werden, was aber einfach durch das Löschen der korrespondierenden Pads von der Basisstation geschehen kann. Alternativ können diese auch nur als *korrumpiert* oder *unbenutzbar* markiert werden.

Wenn bereits benutze OTPs nicht vom Sensorknoten gelöscht worden sind und dieser verloren geht, könnte ein Angreifer diese Pads nutzen um zuvor aufgezeichneten Datenverkehr zu entschlüsseln. Insofern ist es angebracht, die Pads direkt nach Benutzung vom Knoten zu löschen oder aber die verschlüsselten Daten an derselben Speicherstelle zu speichern, an der zuvor das OTP lag.

Die Kritik, dass ein solches System nur eine Lösung zur Verschlüsselung ist, aber keine zur Authentifizierung, gilt nur zur Hälfte. Im vorgestellten Anwendungsfall kann man jedoch nicht von einer *vollständigen und gegenseitigen Authentifizierung* sprechen, da diese dort nur in eine Richtung realisiert ist: Wenn ein empfangender Knoten eine OTP-verschlüsselte Nachricht erfolgreich entschlüsseln kann, impliziert das die Authentifizierung des Senders, da nur er im Besitz des dazugehörigen Pads ist. Da jedoch in diesem Fall keine bidirektionale Kommunikation stattfindet, kann der Sender nicht sicher sein, dass er die Nachricht auch dem beabsichtigten Empfänger zugestellt hat.

Eine gegenseitige Authentifizierung könnte durch ein Challenge-Response-Verfahren realisiert werden, in welchem sich der Empfänger der Nachricht auch durch ein (nur den beiden Kommunikationspartnern bekanntes) OTP ausweist.

Im betrachteten Anwendungsfall wird aber von einer solchen Authentifizierung abgesehen, da zusätzlich auch eine unterbrechungstolerante Kommunikation, welche keine dauerhafte Ende-zu-Ende-Verbindung garantiert, zum Einsatz kommen soll.

Implementierung

Das gerade erläuterte System wurde auf INGA-Sensorknoten (siehe Kapitel 5) und einem Standard-PC als Basisstation (siehe Kapitel 8) implementiert, um es anschließend zu evaluieren.

Sensorknoten des BAN INGA Sensorknoten mit dem Betriebssystem Contiki werden als beispielhafte WBAN-Sensoren eingesetzt. In Abbildung 6.27 ist die grundsätzliche Funktionalität der OTP-Implementierung dargestellt: Zuerst werden die Daten des Sensors aufgenommen. Anhand der bisher verarbeiteten Pads wird die Speicheradresse des aktuellen Pads ermittelt (2.) und das entsprechende Pad in den Arbeitsspeicher des Mikrocontrollers geladen (3.). Nun findet die *XOR*-basierte Verschlüsselung statt (4.).

Alternativ und zur vergleichenden Evaluation kann auch eine AES-Verschlüsselung ausgeführt werden. Dazu wird der jeweils zu verschlüsselnde Block zum Funktransceiver übermittelt (4a.), der ein Hardware-AES-Modul enthält, in welchem dann die AES-Verschlüsselung stattfindet. Die verschlüsselten Daten werden in diesem Fall wieder zurück an den Mikrocontroller übermittelt (4b.).

Anschließend werden die OTP- oder AES-verschlüsselten Daten über den Funktransceiver zur Senke versendet, wobei das μDTN-Protokoll zum Einsatz kommt.

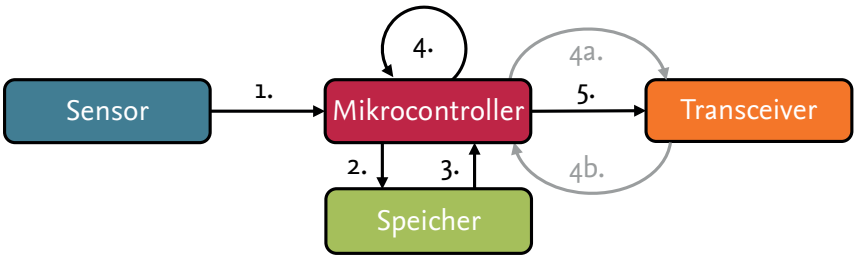


Abbildung 6.27: Umsetzung der OTP- und AES-Implementierung auf den Sensorknoten.

Die Implementierung erlaubt es, auf zwei unterschiedliche Arten von OTP-Speicher zuzugreifen: Für kleinere Datenmengen kann INGAs serieller onBoard-Flash-Speicher verwendet werden, der relativ schnell und energiesparend arbeitet und bis zu 16 MBit an Daten speichern kann. Für größere Mengen sollte auf die SD-Karte zurückgegriffen werden, wobei die augenblickliche Implementierung des Treibers in der Lage ist, bis zu 32 GB zu adressieren. Es werden allerdings jeweils nur Rohdaten gelesen und geschrieben – ein PC-kompatibles Dateisystem wie File Allocation Table (FAT) wurde für diesen Anwendungsfall nicht eingebunden, da die SD-Karte dauerhaft im Knoten verbleibt und somit dieser Overhead gespart werden konnte.

Basisstation Die Basisstation besteht aus einem Standard-PC, an den ein weiterer INGA-Sensorknoten als Gateway zum WBAN angeschlossen ist. Die Software besteht aus einem einfachen Python-Programm, das die OTPs mittels eines Zufallszahlengenerators erzeugt, welcher vom Linux-Betriebssystem zur Verfügung gestellt wird.

Jedes Mal, wenn ein Knoten per USB mit der Basisstation verbunden wird, wird dessen eindeutiger Bezeichner überprüft. Nachdem überprüft wurde, ob dieser zu einem bekannten Knoten gehört, werden die entsprechende Pads dem entsprechenden Knoten zugeordnet und übertragen. Dabei ist es die Aufgabe der Basisstation, über die Zuordnungen zu wachen und die korrespondierenden Pads jeweils entsprechend zu speichern. Da die Identifikation über die Knoten-ID stattfindet, kann dieses System auch mehrere Knoten unabhängig voneinander mit verschiedenen Sets von OTPs versorgen.

Evaluation

Um bestimmen zu können, ob und inwiefern das implementierte System den Anforderungen gewachsen ist, soll es an dieser Stelle evaluiert werden. Dabei wird iterativ vorgegangen: Zunächst wird die Leistungsfähigkeit der Verschlüsselungsalgorithmen quantifiziert, wobei auch gezeigt wird, wie diese sich zur AES-Hardwareverschlüsselung verhalten. Anschließend wird der Durchsatz für den Speicherzugriff bestimmt, wobei hier sowohl der serielle Flash-Speicher als auch der Zugriff auf SD-Karten betrachtet wird. Des Weiteren werden der Kommunikationsoverhead und der Durchsatz auf der Funkverbindung bestimmt, sodass mit der letzten Messung auch gleichzeitig das Gesamtsystem evaluiert ist. Die Ergebnisse sind in Grafik 6.28 am Ende dieses Abschnitts zusammenfassend dargestellt.

Leistungsfähigkeit der Verschlüsselung In Abschnitt 6.2.2 wurde gezeigt, dass die hardwaregestützte AES-Verschlüsselung allen in Software realisierten Algorithmen bei weitem überlegen ist. Während dort (beispielsweise in der Grafik 6.21) allerdings die reine Verschlüsselungsleistung in einem rudimentären System bestimmt wurde, ist bei diesen Messungen auch immer das Gesamtsystem aktiv, was sich

auch auf die Leistungsfähigkeit der AES-Verschlüsselung auswirkt. Um AES- und OTP-Verschlüsselung vergleichend betrachten zu können, wurde die OTP-Padgröße entsprechend der AES-Blockgröße gewählt und beträgt 128 Bit für das evaluierte System. Hierbei war die AES-Hardwareverschlüsselung im Durchschnitt in der Lage 5929 Verschlüsselungsoperationen pro Sekunde durchzuführen, was zu einem Durchsatz von $D_{enc,AES} \approx$759 kBit/s führt.

In derselben Umgebung war die softwarebasierte OTP-Verschlüsselung fähig, 17 689 Verschlüsselungsoperationen pro Sekunde durchzuführen, was wiederum zu einem Verschlüsselungsdurchsatz von $D_{enc,OTP} \approx$2.26 MBit/s führt – auf einem System mit einem Prozessortakt von 8 MHz.

Diese doch beachtlich hohe Geschwindigkeit kommt dadurch zustande, dass die Verschlüsselung auf einfachen *XOR*-Operationen beruht. Eine *XOR*-Operation braucht auf der verwendeten ATmega-Architektur nur einen Taktzyklus, sofern sie zwischen zwei Registern ausgeführt wird. Obwohl das Laden und Speichern dieser Register auch noch wenigstens sechs weitere Taktzyklen benötigt, ist es immer noch um mindestens eine Größenordnung schneller, als der Zugriff auf einen SPI-Bus, über welchen die AES-Verschlüsselung dann zu einer externen Hardwarekomponente ausgelagert wird.

Für den betrachteten Anwendungsfall heißt das also, dass die Leistungsfähigkeit der reinen Verschlüsselungsoperationen mehr als ausreichend ist, da die Daten weitaus schneller verschlüsselt werden können, als sie erzeugt werden:

$$D_{gen} = 1.5\,\text{kilobit/s} \ll 2.26\,\text{megabit/s} = D_{enc,OTP} \tag{6.6}$$

Speicherzugriff Wenn große Mengen an OTPs benötigt werden, müssen diese natürlich auch von einem geeignetem Speicher in ausreichender Geschwindigkeit zuverlässig eingelesen werden können. INGAs Mikrocontroller verfügt über 128 kByte integrierten Flash-Speicher, welcher jedoch hauptsächlich durch das Betriebssystem und die Anwendungen belegt wird. Gerade bei größeren Mengen muss deshalb auf externen Speicher zurückgegriffen werden. INGA verfügt dabei zum einen über einen relativ schnellen onBoard-Flash-Speicher, der über ein Dual-Buffer-Interface an den SPI-Bus des Mikrocontrollers angebunden ist. Zum anderen können quasi beliebig große Speicher als microSD-Karten angesprochen werden, wobei der aktuelle Treiber 32 GB adressieren kann. INGAs microSD-Kartenslot ist allerdings über ein kompatibles – aber langsames – SPI-Protokoll angebunden, welches fast die meisten SD-Karten aus Kompatibilitätsgründen unterstützen. Die Benutzung eines PC-kompatiblem Dateisystems wie z.B. FAT ist prinzipiell zwar beispielsweise mit der RATFAT-Implementierung [140] möglich, jedoch für diesen Anwendungsfall nicht notwendig – das direkte Lesen und Schreiben von Blöcken ist in diesem Fall völlig ausreichend.

Wie bereits in Abschnitt 6.2.4 erwähnt, wurde eine feste Blockgröße für OTPs von 128 Bit festgelegt, um direkt mit AES vergleichbar zu sein. Vom onBoard-Flash konnten bei dieser Größe ∼3541 Pads/s gelesen werden; das entspricht einem lesenden Durchsatz von $D_{mem,flash} \approx 453.3\,\text{kBit/s}$.

SD-Karten verfügen in der Regel über eine feste Seitengröße von 512 Byte; es ist also sinnvoll, auch immer 512 Byte auf einmal zu lesen, was 32 OTPs mit 128 bit Größe entspricht. In dieser Konfiguration war es möglich ∼20 Seiten/s zu lesen, was ∼630 Pads/s entspricht. Das wiederum führt bei der SD-Karte zu einem Speicherdurchsatz von $D_{mem,SD} \approx 80.7\,\text{kBit/s}$. Das sieht zwar auf den ersten Blick nicht beeindruckend aus, ist aber aufgrund der langsamen Anbindung des Speichers durchaus in der erwarteten Größenordnung und entspricht in etwa den Messwerten bei der Evaluation von INGA (siehe Grafik 5.16 in Abschnitt 5).

Nichtsdestotrotz: Solange die durch die Sensorik generierte Datenrate unterhalb der durchschnittlichen Datenrate für den Speicherzugriff liegt, wird das System auch mit dem relativ langsam angebundenen SD-

Speicher zufriedenstellend funktionieren. Das ist nicht nur für den gerade diskutierten Anwendungsfall (mit 50 Hz abgetastete Accelerometerdaten) gültig, sondern auch für etliche weitere Anwendungsfälle, da selbst im Falle des langsam angebundenen SD-Speichers noch > 53 mal mehr OTP ausgelesen werden könnten.

$$D_{gen} = 1.5\,\text{kilobit/s} \ll 80.7\,\text{kilobit/s} = D_{mem,SD} \ll 453.3\,\text{kBit/s} = D_{mem,flash} \tag{6.7}$$

Kommunikationsoverhead Im Gegensatz zu allen in Abschnitt 6.2.3 erwähnten und in Abbildung 6.23 dargestellten Umsetzungen auf Sicherungs-, Vermittlungs- und Transportschicht, erzeugt die Lösung mit One-Time-Pads keinen weiteren Kommunikationsoverhead – sofern die OTPs genau so lang wie der zu verschlüsselnde Klartext gewählt werden: Ist das der Fall, verlängert eine ausschließende Disjunktion diese Nachricht nicht. Im Standardfall wirkt sich also die Verwendung von OTPs nicht auf die Menge der zu übertragenden Daten aus.

Will man allerdings gegenüber einem eventuellen Angreifer auch die Länge der übermittelten Nachrichten verheimlichen, kann *Padding* (Auffüllen) angewendet werden: Zur Verschleierung der Klartextlänge kann es zum Beispiel sinnvoll sein, jeweils die volle Nutzlastgröße auszunutzen und so Nachtrichten mit immer derselben Größe zu schicken – unabhängig von der tatsächlichen Länge des Inhalts. Dazu werden einfach die nicht genutzten Bytes aufgefüllt – in der Regel nach einem speziellen (und natürlich ebenfalls verschlüsselten) Zeichen. Auf diese Weise entsteht dann natürlich auch ein Kommunikationsoverhead, aber dieser ist nicht OTP-spezifisch, da sich Padding auch für jede Verschlüsselungsmethode zur Verschleierung der Nachrichtenlänge anwenden lässt.

Datenübertragung Die Datenübertragung wurde mittels μDTN (siehe Abschnitt 6.1) realisiert, wobei eine feste Nutzlastgröße von 80 Byte gewählt wurde. Diese Größe ergibt sich primär aus der Tatsache, dass die implementierte OTP-Verschlüsselung mit Blöcken von 128 Bit = 16 Byte Größe arbeitet und bei vollständiger Ausnutzung der zur Verfügung stehenden Nutzlast unvollständige Blöcke übermittelt werden würden. So konnten also in einem μDTN-Bundle insgesamt 5 OTP-verschlüsselte Blöcke übermittelt werden. Da also einige Bytes im Vergleich zu den in Abschnitt 6.1.4 erwähnten maximalen Nutzlastgröße von 94 ungenutzt blieben, ergab sich ein gemessener durchschnittlicher Durchsatz von $D_{transmit} = 30.72\,\text{kilobit/s}$. Diese obere Grenze ist aber auch hier dem Empfänger (siehe auch Abschnitt 6.1.4) geschuldet, der nicht in der Lage war, mehr Daten über die serielle Verbindung zuverlässig zum Hostsystem zu übertragen.

$$D_{gen} = 1.5\,\text{kilobit/s} \ll 30.7\,\text{kilobit/s} = D_{transmit} \tag{6.8}$$

Das heißt, dass die unterbrechungstolerante Datenübertragung die erzeugten Daten sicher zustellen kann. Theoretisch könnte auf diese Weise auch das zwanzigfache Datenaufkommen transportiert werden, aber um noch von der Unterbrechungstoleranz profitieren zu können, sollte dieses Limit nicht komplett ausgereizt werden.

Speicherbedarf Aufgrund ihrer Simplizität hat die reine OTP-Verschlüsselung nur einen sehr geringen Speicherbedarf von 380 Byte Programmspeicher (ROM) bzw. 2 Byte Arbeitsspeicher (RAM), wenn der onBoard Flash-Speicher als OTP-Speicher dient. Fairerweise sollte man den Flash-Treiber bei dieser Berechnung ebenfalls berücksichtigen, da dieser ja für den Speicherzugriff nötig ist und nicht in allen Anwendungsfällen gebraucht wird. Unter Berücksichtigung des zusätzlich zu ladenden Flash-Treibers ergibt sich dann ein Gesamtspeicherbedarf von 2222 Byte ROM bzw. 38 Byte RAM. In Tabelle 6.3 ist der Speicherbedarf der OTP-Verschlüsselung aufgeschlüsselt; hier ist auch der Overhead für die Nutzung

von SD-Karten als Speichermedium angegeben, welcher sich jeweils als etwas höher als bei der Nutzung des Flashs darstellt.

Tabelle 6.3: Speicherbedarf der OTP-Implementierung bei Flash- und SD-Speicheranbindung.

	ROM (kByte)			RAM (Byte)		
	Gesamt	Differenz	zu Treiber	Gesamt	Differenz	zu Treiber
Grundsystemsystem	47.19	-	-	2093	-	-
... mit Flash-Treiber	49.03	1.84	-	2129	36	-
... und OTP-Unterstützung	49.41	2.22	0.38	2131	38	2
... mit SD-Treiber	50.99	3.81	-	2205	76	-
.. und OTP-Unterstützung	51.52	4.34	0.53	2207	78	2

Im untersuchten und evaluierten Anwendungsfall ist die erzeugte Datenrate in jedem Fall weitaus geringer als alle beteiligten Komponenten zu verarbeiten in der Lage wären:

$$D_{gen} \ll D_{transmit} \ll D_{mem,SD} \ll D_{mem,flash} \ll D_{enc,OTP} \tag{6.9}$$

Die Evaluation der einzelnen Komponenten und des Gesamtsystems konnte zeigen, dass das System stabil und mit durchaus annehmbarer Leistung läuft.

In Abbildung 6.28 ist der gemessene Durchsatz für die einzelnen Komponenten im Vergleich aufgetragen. Es lässt sich erkennen, dass auch für Anwendungsfälle mit höheren Datenraten noch Kapazitäten frei sind.

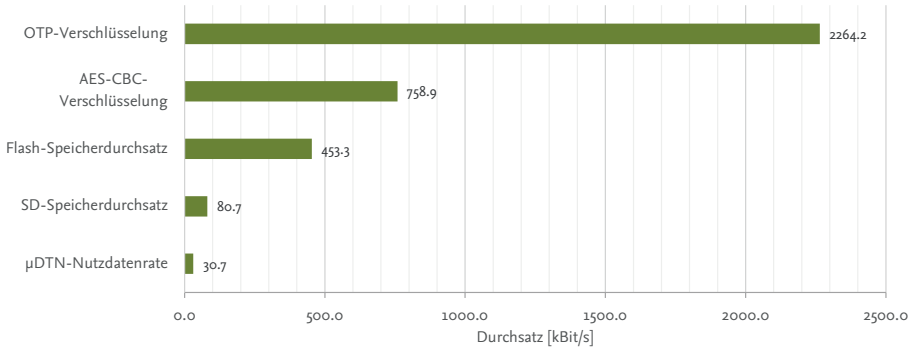


Abbildung 6.28: Der Durchsatz der einzelnen Komponenten für das OTP-Verschlüsselungssystem.

6.2.5 Zusammenfassung: Sicherheit und Verschlüsselung

Es steht außer Frage, dass Daten, die am Körper eines Menschen aufgezeichnet werden, persönlich und als solche schützenswert sind. Gerade bei der drahtlosen Übertragung dieser Daten muss deshalb auf ein hohes Maß an Sicherheit geachtet werden: Krankenversicherungsunternehmen könnten sich für Blutdruck und Herzfrequenz interessieren, Diebe für die Trainingsgewohnheiten beim Outdoor-Sport und potentielle Sexualpartner für die Fruchtbarkeit [141]. Eine starke Verschlüsselung dieser Daten ist deshalb nicht nur wünschenswert, sondern prinzipiell notwendig.

In diesem Abschnitt wurden verschiedene Ansätze und Verfahren zur Verschlüsselung der drahtlosen Datenübertragung vorgestellt. Alle bisher propagierten Verfahren haben ihre Stärken und Schwächen: Manche sind einfach anzuwenden, vertrauen dafür aber auf gemeinsame und vorab verteilte Schlüssel; andere verwenden aufwendige Verfahren zum Schlüsselaustausch, benötigen dafür aber einiges an Rechenkapazität.

Allen Verfahren gemein ist, dass sich die Algorithmen und Verfahren zur Verschlüsselung nicht problemlos austauschen lassen; sei es, weil sie in Hardware implementiert sind, oder weil die Rechenleistung der leistungsschwachen Sensorknoten für das Verschlüsseln mit verdoppelter Schlüssellänge nicht ausreicht.

Mit der Verwendung von One-Time-Pads wurde eine Möglichkeit vorgestellt, wie man mit wenig Aufwand ein besonders sicheres System aufbauen kann, welches zumindest kryptographisch nicht zu brechen ist. Die Menge an günstig verfügbaren Speicher lässt gerade für Szenarien, in denen nur wenige Daten anfallen, eine große Reserve für eine zeitlich ausgedehnte sichere Verschlüsselung mit OTPs. Auch für anderen Szenarien, wie beispielsweise die Übertragung sicherheitsrelevanter Messwerte bei der Überwachung einer Raffinerie [45], lässt sich dieses Verfahren adaptieren – in diesem Fall würde man aber geschickterweise die Verteilung neuer Schlüssel durch das Austauschen der SD-Karten realisieren, welche beispielsweise in einem Zug mit den Batterien ausgewechselt werden könnten.

Die Evaluation dieses Systems zeigt, dass es auch in der Praxis funktioniert und dass die erreichbaren Datenraten für die anvisierten Anwendungsfälle mehr als ausreichend sind.

6.3 Datenreduktion

In den vorangegangenen Abschnitten wurde aufgezeigt, dass das vorgestellte System zur Datenübertragung und zur Verschlüsselung, einem Großteil der Anwendungsfälle gerecht wird. Wenn aber die aufgenommene Datenmenge zu groß wird, um sie speichern oder versenden zu können, müssen geeignete Maßnahmen getroffen werden, um die Funktionalität des Gesamtsystems weiterhin zu gewährleisten.

Naheliegende Ansätze sind Kompression und Aggregation der Daten auf den Sensorkonten oder im Sensornetzwerk. Bei der Aggregation werden dazu Messwerte zusammengefasst: Wenn beispielsweise der Blutdruck über eine lange Zeit aufgezeichnet wird und der verfügbare Speicher zur Neige geht, könnten z.B. für bestimmte Zeitabschnitte nur noch Mittelwerte anstelle der Rohdaten gespeichert werden. Bei sekündlicher Abtastung könnte man das Datenaufkommen beträchtlich verringern. Würde man die 3600 Messwerte einer Stunde zu einem einzigen Mittelwert zusammenfassen, ergäbe sich eine Ersparnis von $> 99.99\,\%$. Selbst wenn man zusätzlich noch die Minima und Maxima speicherte, bliebe die Kapazitätsersparnis in dieser Größenordnung. Was zunächst unschlagbar klingt, muss bei näherer Betrachtung allerdings relativiert werden. So müssen bei solch einer Aggregation alle gespeicherten Daten noch einmal gelesen und dann der Mittelwert aus ihnen berechnet werden, was einen nicht unerheblichen Aufwand darstellt. Dem könnte man zwar entgegenwirken, indem man schon während der Aufzeichnung auch gleich die Mittelwertbildung betreibt, aber auch das hilft wenig gegen den zweiten Punkt:

Die verlustbehaftete Datenspeicherung ist schlicht nicht gewünscht! In zahlreichen Gesprächen mit den im GAL-Projekt beteiligten Wissenschaftlern und Ärzten, die diese Daten dann auswerten, wurde klargestellt, dass man am liebsten *alle Rohdaten* zur späteren Auswertung zur Verfügung hätte – und zwar am besten *lückenlos* über den gesamten Aufzeichnungszeitraum. Die Anforderung „lückenlos“ führte letztendlich zur Adaption der DTN-Ansätze, die Anforderung, auch alle Rohdaten zu bekommen, spricht letztendlich gegen jegliche Art von Aggregation; gleichzeitig schließt sie auch jede Art von *verlustbehafteter* Kompression aus.

Insofern sollen im Folgenden vorwiegend Mechanismen zur *verlustfreien* Datenreduktion diskutiert werden. Vorab soll jedoch ein Überblick über mögliche Ansätze zur Datenreduktion gegeben werden. Anschließend wird deshalb analysiert, ob Teile der Informationen redundant bzw. überflüssig sind. Danach wird anhand von Langzeitmessungen bestimmt, wie sich die aufgezeichneten Daten zueinander verhalten; also, in welchem Wertebereich sie liegen und ob es Abhängigkeiten gibt, die sich für eine spätere Komprimierung ausnutzen lassen. Am Ende des Abschnitts folgt eine Evaluation der einzelnen Ansätze.

6.3.1 Ansätze zur Datenreduktion

Um zu verstehen, wo Möglichkeiten zur Reduktion des Datenaufkommens existieren, sollte man sich zunächst klar darüber werden, wie die Daten entstehen und welche Systeme an diesem Prozess beteiligt sind. In Abbildung 6.29 ist deshalb zunächst exemplarisch ein Modell der Datenaufnahme gezeigt: Am linken Rand ist die zu quantifizierende physikalische Welt, deren Messgrößen zunächst in elektronisch bestimmbare Größen umgewandelt werden müssen. Am rechten Rand stehen dann die Daten zur Speicherung und/oder Übertragung bereit; der Messaufnehmer und die AD-Umsetzung sind obligatorisch, alle anderen Blöcke sind optional.

Mit Rücksicht auf die in Abschnitt 5.1 erläuterte Terminologie werden in diesem Abschnitt die *Messaufnehmer* nicht als Sensoren bezeichnet; *Sensoren* sind also weiterhin als *Bauteile* zu verstehen: *Analoge Sensoren* beinhalten demnach den Messaufnehmer und gegebenenfalls analoge Filter und eine Messbereichsanpassung. *Digitale Sensoren* beinhalten zusätzlich den ADC und können digitale Filter enthalten.

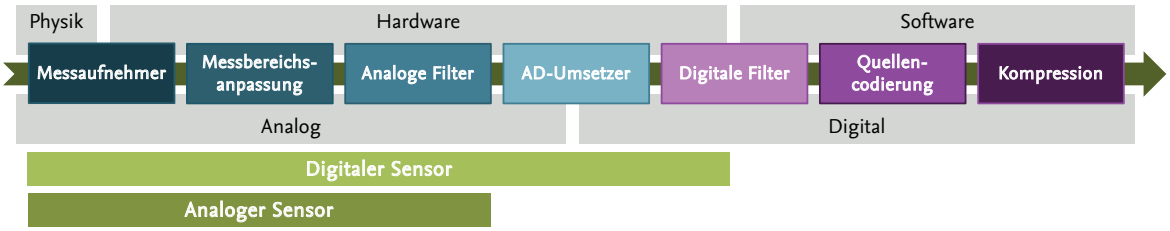


Abbildung 6.29: Die einzelnen Schritte vom physikalischen Ereignis (links) bis zu den übertragbaren Daten.

Der Messaufnehmer wandelt also die physikalische Größe in eine messbare Größe um. Ein optionaler analoger Filter kann diese Signale beispielsweise glätten und so bestimmte unerwünschte Effekte (wie z.B. hohe Frequenzen) herausfiltern. Die Analog-Digital-Umsetzung quantifiziert das analoge Signal zeit- und wertdiskret. Nun können optionale digitale Filter (in Hardware oder Software) angewendet werden. Nach einer ebenfalls optionalen Quellcodierung kann eine ebenso optionale Kompression erfolgen, bevor die Daten dann weiterverarbeitet werden.

Messaufnehmer

In den wenigsten Fällen lässt sich dabei eine Größe direkt messen, vielmehr misst man ihre Auswirkung auf eine (elektrisch) detektierbare Größe. Siehe dazu auch Abschnitt 4.1.2, in dem die Funktionsweise eines MEMS-Accelerometers kurz erklärt wird.

Detektierbare Größen sind dabei *Spannung*, *Strom*, *Widerstand*, *Induktivität*, *Kapazität* oder auch Frequenz, wobei letzteres streng genommen nur die Änderung anderer Größen ist. Die im Bereich der Mikrocontroller am häufigsten anzutreffende Umwandlung ist die in eine Spannung, da diese mit den integrierten ADCs einfach abgetastet und quantifiziert werden kann, weswegen beim Vorhandensein anderer elektrisch bestimmbarer Größen diese in der Regel (beispielsweise mit einer Messbrücke) in eine Spannung umgewandelt werden.

Am einfachsten misst man beispielsweise eine Temperatur, indem man sich zu Nutze macht, dass bestimmte leitende Stoffe ihren elektrischen Widerstand mit der Temperatur ändern. Wenn man nun eine Spannung an ein System, bestehend aus diesem veränderlichen Widerstand und einem weitestgehend konstanten Widerstand (Spannungsteiler), anlegt, erhält man eine messbare Spannung, die sich mit der Temperatur verändert.

Analoger Filter

Bevor eine Größe zu einer Spannung gewandelt wird, kann sie bereits gefiltert werden. Wenn man beispielsweise einen Beschleunigungssensor in eine Flüssigkeit mit hoher Viskosität (z.B. Öl) legt, werden die darauf wirkenden Beschleunigungen und damit auch die Messwerte gedämpft.

Aber auch nach der Umwandlung in eine Spannung lassen sich die Messwerte analog (vor-)filtern. Mit einer einfachen Widerstand-Kondensator-Kombination (RC-Glied) lässt sich z.B. ein Tiefpassfilter realisieren. Als einen Tiefpass bezeichnet man dabei einen Filter, der die Signale bis zu einer bestimmten Grenzfrequenz nahezu ungedämpft passieren lässt; Signale höherer Frequenz werden deutlich abgeschwächt. Bei der Analyse des Actigraph (siehe Abschnitt 3.1.2) konnte beispielsweise festgestellt werden, dass der dort verwendete analoge Beschleunigungssensor mit einem Tiefpass von $f_{BW} = 24.85\,\text{Hz}$ für jede der drei Achsen versehen wurde, bevor die Spannungen dann vom prozessorinternen ADC in digitale Werte gewandelt wurden. Die analoge Filterung führt in jedem Fall zu einer Beeinflussung der gemessenen Werte: Im gerade beschriebenen Beispiel werden also Frequenzen $> 25\,\text{Hz}$ gar nicht erst an die weiteren Verarbeitungsschritte „weitergereicht“. Das kann je nach Anwendungsfall durchaus gewollt sein, jedoch sollte man dann auch den Rest des Systems darauf auslegen: Unter Einhaltung des Nyquist-Shannon-Abtasttheorems [77] würde in diesem Fall eine Abtastung mit 50 Hz noch Sinn ergeben, eine Abtastung mit mehr als 50 Hz würde jedoch einfach nur mehr Daten produzieren, ohne zu einem Informationsgewinn zu führen.

Analoge Filter können bei analogen Sensoren in der Regel durch eine externe Beschaltung realisiert werden, teilweise sind sie aber auch „ab Werk“ mit einem festen unveränderlichen analogen Filter ausgestattet. Bei digitalen Sensoren lassen sich die Filter teilweise durch das Setzen bestimmter Registerwerte konfigurieren. Analoge Filter haben einen direkten Einfluss auf die gemessenen Werte: Signalanteile, die mit einem analogen Filter herausgefiltert wurden, sind unwiederbringlich verloren.

Messbereichsanpassung

Manche Sensoren lassen sich in unterschiedlichen Wertebereichen betreiben. Die Wahl des Messbereichs hat dabei manchmal auch eine direkte Auswirkung auf die Auflösung (s.u.). So kann beispielsweise das in INGA verwendete Accelerometer (siehe Abschnitt 5.2.4) in insgesamt vier verschiedenen Messbereichen betrieben werden ($\pm 2\,\text{g}$, $\pm 4\,\text{g}$,$\pm 8\,\text{g}$ und $\pm 16\,\text{g}$). Im speziellen Fall wirkt sich die Wahl des Messbereichs auch direkt auf die Auflösung und damit auf die erzeugte Datenmenge aus. So variiert die Auflösung zwischen 10 Bit (Messbereich $\pm 2\,\text{g}$) und 13 Bit (Messbereich $\pm 16\,\text{g}$) pro Achse. Für den Anwender hat das den Vorteil, dass die Sensitivität über den gesamten Messbereich konstant bleibt, da so immer 256 Werte pro g unterschieden werden können. Bei manchen anderen Sensoren bleibt die Auflösung bei höheren Messbereichen gleich, was zu einer Verschlechterung der Sensitivität führt.

Wird der Messbereich zu klein gewählt (beispielsweise $\pm 2\,\text{g}$) und es treten größere Werte auf ($< -2\,\text{g}$ oder $> 2\,\text{g}$), führt das dazu, dass die Daten für den fälschlicherweise angenommenen Maximalwert auch für alle Werte darüber ausgegeben werden. Im Beispiel würden also alle Beschleunigungen $\geq 2\,\text{g}$ durch das Datum 1024 und alle Beschleunigungen $\leq -2\,\text{g}$ durch 0 repräsentiert werden. Bei einem zu unrealistisch gewählten Messbereich mit Werten, die nicht erreicht werden können, wie beispielsweise mehr als $\pm 4\,\text{g}$ für Menschen in normalem Umfeld (also außerhalb eines Kampfjets), wird Messbereich verschwendet und Raum für Daten gelassen, der nie genutzt werden wird.

Generell sollte also der Messbereich immer auf den erwarteten Messbereich eingestellt werden – sei es um eine höhere Sensitivität zu erhalten oder um Datenaufkommen zu sparen.

Analog-Digital-Wandlung

Bei der Analog-Digital-Wandlung geht es also darum, aus analogen elektrischen Spannungen, digitale und mit dem Rechner verarbeitbare Werte zu erzeugen. *Digital* heißt dabei auch immer *diskret* – und zwar zu

bestimmten Zeiten (zeitdiskret) und in bestimmten Abständen (wertediskret). Bei dieser Wandlung werden bereits wichtige Parameter bestimmt, die das spätere Datenaufkommen maßgeblich beeinflussen.

Abtastrate Die Abtastrate bestimmt, wie oft ein Wert abgetastet wird – wobei diese Frequenz je nach Sensor und je nach Anwendungsfall sehr unterschiedlich sein kann. Wie bereits erwähnt, ergibt eine Abtastung mit mehr als der für die Einhaltung des Nyquist-Shannon-Abtasttheorems nötigen doppelten Grenzfrequenz eines eventuell vorhandenen Tiefpasses wenig Sinn. Um Datenaufkommen zu vermeiden, ist es also generell sinnvoll, die Abtastrate nach der Beschaffenheit der erwarteten Messwerte festzulegen. In Abschnitt 4.2 wurden bereits mögliche und typische Abtastraten aufgezeigt, wobei zwischen dem was für einen Anwendungsfall sinnvoll ist und dem was technisch möglich ist, durchaus mehrere Größenordnungen liegen können.

Auflösung Die Anzahl der zur Verfügung stehenden diskreten Quantisierungsstufen ist durch die Auflösung bestimmt. Eine Auflösung von beispielsweise 10 Bit bedeutet, dass für den Wertebereich $2^{10} = 1024$ unterschiedliche Stufen zur Verfügung stehen, wobei ein zusätzliches Bit einer Verdoppelung des Wertebereichs gleichkommt ($2^{11} = 2048$). Auch hier sollte man sich am Benötigten orientieren: Wenn Temperaturen im Bereich zwischen 34.0 und 43.0 °C mit einer Genauigkeit von 0.1 °C aufgenommen werden sollen, reichen dafür 90 verschiedene Messwerte aus; eine sinnvolle Auflösung wäre demnach 7 Bit ($2^7 = 128$).

Bei den meisten digitalen Sensoren oder AD-Wandlern werden die Ergebnisse in ganzen Bytes ausgegeben. So wird z.B. ein 10 Bit Messwert in 2 Byte repräsentiert, wobei 3/4 des zweiten Bytes ungenutzt bleiben – in diesem Fall würde auch eine gewählte Auflösung von 16 Bit zunächst nicht mehr an Speicher belegen. Es empfiehlt sich also entweder innerhalb von Bytegrenzen zu bleiben oder anschließend eine Quellcodierung vorzunehmen (siehe Abschnitt 6.3.1).

Wertebereich Wenn analoge Sensoren betrieben werden, so ist auch auf den erwarteten Spannungsbereich zu achten, wobei sich das am besten an einem Beispiel erläutern lässt – ebenfalls bei der Analyse des Actigraph konnte Folgendes festgestellt werden: Das analoge Accelerometer ist an den ADC des Prozessors angeschlossen, welcher eine Auflösung von 12 Bit bietet, woraus sich $2^{12} = 4096$ verschiedene mögliche Werte über den gesamten Messbereich ergeben. Der Messbereich umfasst im Beispiel 12 g (−6 g bis 6 g), was zu der vereinfachten Annahme verleitet, die einfache Division von $4096/12 \approx 341$ würde zu der Anzahl der Werte pro g führen (LSB/g). Der ADC arbeitet zwar linear, jedoch im Bereich zwischen 0 V (bzw. GND) und einer wählbaren Referenzspannung. Die Referenzspannung konnte durch Messung auf $V_{Ref} = V_{CC} = 3\,\text{V}$ bestimmt werden. Die 4096 möglichen Werte für die AD-Wandlung teilen sich also auf den Spannungsbereich von 0 V bis 3 V auf, wobei die erwarteten Messwerte laut Datenblatt des Sensors nur im Bereich zwischen 0.3 V bis 2.7 V liegen. Das heißt, dass 0.6 V des abgetasteten Bereichs zwischen 0 V und 3 V nicht nutzbar sind – damit sind im Beispiel 20 % des Wertebereichs quasi verschwendet, was sich in diesem Fall jedoch nicht auf die Bandbreite sondern nur auf die Sensitivität auswirkt. Würde die Diskrepanz zwischen Abtastbereich der AD-Wandlung und dem Spannungsbereich eines analogen Sensors noch größer sein, könnte eine bessere Wahl des Bereichs bei gleichbleibender Sensitivität auch wiederum Bandbreite sparen.

Digitale Filter

Digitale Filter können entweder im Digitalteil eines digitalen Sensors, auf speziellen Bauteilen (beispielsweise Digitaler Signalprozessor (DSP)) oder in der Software des verarbeitenden Mikrocontrollers realisiert sein. Im Gegensatz zur analogen Filterung werden hier Funktionen auf den bereits quantisierten und abgetasteten Werten ausgeführt. So können beispielsweise auch Glättungsfunktionen und eine gleitende Mittelwertbildung

realisiert werden. Die Ergebnisse des digitalen Filterns sind auch irreversibel und führen zum unwiederbringlichen Verlust von Signalanteilen.

Quellenkodierung

Die Quellencodierung reduziert redundante Informationen der aufgenommenen Daten, ohne die enthaltende Information zu verändern.

Ein Beispiel ist die Huffman-Kodierung [142], die als Entropiekodierung das Wissen über die Auftrittswahrscheinlichkeit bestimmter Symbole nutzt, um so besonders häufig auftretende Symbole mit wenig Bits zu kodieren und seltener auftretende Symbole mit entsprechend mehr Bits darzustellen. Essentiell dafür ist allerdings, dass die Auftrittswahrscheinlichkeit bekannt ist, denn eine schlechte/falsche Konfiguration der Huffman-Kodierung kann auch zu einer Vergrößerung der Datenmenge führen.

Die adaptive Variante der Huffman-Kodierung baut den Kodierbaum entsprechend der realen Auftrittshäufigkeit der einzelnen Symbole auf. Allerdings muss dann auch der (sich ändernde) Baum den beteiligten Kommunikationspartnern bekannt gemacht werden, damit diese die empfangenen Daten auch dekodieren können. In [143] wurde eine solche adaptive Huffman-Kodierung für WSNs umgesetzt – allerdings mit einer sehr begrenzten Anzahl von Symbolen. Ein Huffman-Baum mit nur 256 Symbolen benötigt auf einem Mikrocontroller demnach 4 kByte Arbeitsspeicher – bei größeren Bäumen entsprechend mehr.

Kompression

In [144] werden einige Ansätze zur verlustfreien Kompression in WSNs miteinander verglichen. Die Autoren stellen fest, dass klassische Kompressionsalgorithmen, wie LZ77 [145], LZW [146] oder auch die gerade erwähnte adaptive Huffman-Kodierung, bei fehlerbehafteten Übertragungskanälen ab einer gewissen Fehlerrate ($>$ 10 %) schlechter funktionieren, als wenn gar keine Kompression zum Einsatz käme. Da diese Algorithmen beim Dekomprimieren Abhängigkeiten zu zuvor dekomprimierten Daten haben, ist es auch nicht weiter verwunderlich, dass sich (ähnlich wie schon bei der CBC-Verschlüsselung in Abschnitt 6.2.1) wenn diese Daten nicht vorhanden sind, zukünftige Daten auch nicht dekomprimieren lassen. Auf die normale (nicht-adaptive) Huffman-Kodierung trifft das nicht zu, da hier die statischen Kodierbäume sowohl dem Sender als auch dem Empfänger bekannt sind und so jedes Symbol unabhängig von vorherigen Übertragungen dekomprimiert werden kann. Die Grenzen der Übertragung müssen aber auch hierfür eindeutig identifizierbar sein: Es ist aber auch generell davon abzusehen, einzelne Symbole über Paket- oder Rahmen-Grenzen hinweg zu kodieren, da sich sonst auch hier Fehler fortpflanzen können.

Das Problem der fehlerbehafteten Übertragungskanäle wird in [144] dadurch gelöst, dass empfangene Pakete zusammengefasst bestätigt werden und sich nur der Inhalt der bestätigten Pakete auf die Anpassung der zur Kompression verwendeten Wörterbücher auswirkt.

Verlustbehaftete Kompression kommt in der Regel bei der Stand- und Bewegtbildkompression zum Einsatz [147]. Die in der Regel geringen Datenübertragungsraten von WSNs würden auch sonst für eine Übertragung solcher Daten nicht ausreichen. Auch bei Audio-Daten wird eine verlustbehaftete Kompression eingesetzt [148].

Eine Aggregation von Daten kann, wie eingangs erwähnt, verlustbehaftet sein, es können bei mehreren ähnlichen aufgezeichneten Daten innerhalb eines Netzwerks auch verlustfrei Aggregationen angewendet werden [149].

Zusammenfassung: Möglichkeiten zur Dateneinsparung bei der Signalverarbeitung

Eine Veränderung der Parameter der Messaufnehmer, der Messbereichsanpassung, der analogen Filter, der AD-Umsetzer und der digitalen Filter verändert das weiterzuverarbeitende Signal in jedem Fall. Nichtsdestotrotz bietet sich hier auch ein großes Einsparpotential. Gerade die Abtastrate der Analog-Digital-Wandlung sollte deshalb sinnvoll und dem Anwendungsfall angepasst gewählt werden.

Während analoge wie digitale Filter immer verlustbehaftet arbeiten, geht bei der Quellenkodierung keinerlei Information verloren. Eine Kompression lässt sich sowohl verlustbehaftet als auch verlustfrei realisieren, wobei normale Kompressionsalgorithmen bei der Dekompression unter dem Einfluss verlorener Pakete dieselben Probleme aufweisen, wie auch Verschlüsselungsalgorithmen, die auf das Vorhandensein zuvor übermittelter Daten aufbauen.

6.3.2 Zustandslose Quellenkodierung

Da eine statistikbasierte Quellenkodierung wie die Huffman-Kodierung nur bei Kenntnis der erwarteten Messwerte sinnvoll angewendet werden kann, soll an dieser Stelle zunächst eine einfache und zustandslose Quellenkodierung zum Einsatz kommen. Sie wird deshalb als *zustandslos* bezeichnet, da sie keinerlei Abhängigkeiten aufweist – weder von Symbolauftrittswahrscheinlichkeiten noch von vorherigen Messwerten oder dem Systemzustand.

Alle folgenden Betrachtungen werden beispielhaft für die auf dem (in Kapitel 5 vorgestellten) Sensorknoten INGA vorhandenen Sensoren durchgeführt. Bei anderen und/oder weiteren Sensoren können dieselben Techniken angewandt werden. Die konkreten Umsetzungen müssen aber für den Einzelfall angepasst werden.

Zunächst werden die vorhandenen Sensoren auf INGA und ihre Repräsentation der Daten analysiert. Anschließend wird ein ebenso einfacher wie effizienter Ansatz zur Verringerung dieser Daten präsentiert, der unabhängig von der Art der übermittelten Daten funktioniert.

Beschleunigungssensor - ADXL345

Der in INGA zum Einsatz kommenden ADXL345 liefert Rohdaten für die Beschleunigungswerte jeweils als 16-Bit-Werte pro Achse, also $3 \cdot 16\,\text{Bit} = 6\,\text{Byte}$ pro Abtastzeitpunkt. Jeder dieser 16-Bit-Werte enthält, je nach Konfiguration, jedoch nur bis zu 13 Bit an Information -– die übrigen 3 (bis 6) Bit sind ungenutzt (siehe Abbildung 6.30).

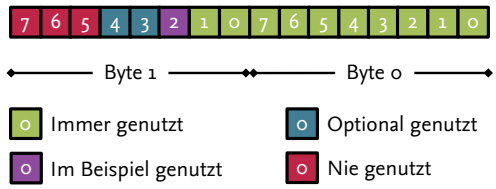


Abbildung 6.30: Die vom Beschleunigungssensor für eine Achse ausgegebenen Daten.

Für den Fall, dass die genutzten (bis zu) 13 Bit gleichverteilt sind, ergibt sich also eine Entropie von 13 Shannon, welche im konkreten Fall mit 16 Bit abgebildet werden. D.h. lediglich 81.25 % (13/16) des möglichen Informationsgehalts wird genutzt.

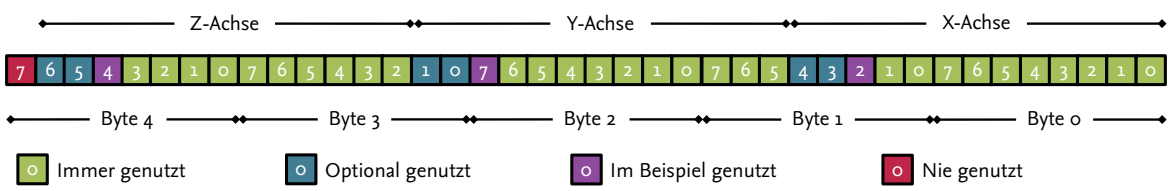


Abbildung 6.31: Fünf Byte für die Daten der drei Achsen des Beschleunigungssensors.

Durch eine geschickte Quellcodierung dieser Daten (siehe Abbildung 6.31) lässt sich dieses Datenaufkommen einfach reduzieren. Nach erfolgter Quellenkodierung sind somit nur noch 5 Byte pro Abtastzeitpunkt nötig.

Dabei bleibt zwar noch immer 1 Bit ungenutzt, aber der Informationsgehalt steigt auf 97.5 % (39/40).

Da in der Informationsverarbeitung auf dem Mikrocontroller Bytes zum Einsatz kommen, kann man diese Darstellung als hinreichend ansehen. Um jedoch eine maximale Entropie und einen Informationsgehalt von 100 % zu erreichen und keinen Overhead durch ungenutzte Bits zu verschwenden, müsste man die Daten mehrerer Abtastzeitpunkte zusammenfassen. Im konkreten Fall könnte man den Overhead auf 0 reduzieren, wenn man 8 Abtastintervalle (mit je 39 Bit) in 39 Byte (312 Bit) zusammenfassen würde. Dieses hätte jedoch den Nachteil, dass man relativ aufwendige Strukturen vorhalten muss und die Daten nicht direkt nach dem Aufzeichnen weiterverarbeiten kann, sondern diese nur in Blöcken von jeweils 8 Intervallen zur Verarbeitung zur Verfügung stehen würden. Mit der präferierten Lösung, die Messwerte für die drei Achsen jeweils mit 5 Byte darzustellen (anstelle von 6 Byte zuvor) gibt sich eine Reduktion auf 5/6, also auf 83.33 % der vorherigen Datenrate.

Gyroskop - L3G4200D

Beim Gyroskop haben alle abgetasteten Werte eine Auflösung von 16 Bit, was genau 2 Byte entspricht. Insofern ist hier eine einfache Reduktion durch geschickteres Verteilen der Messwerte auf Bytes nicht gegeben.

Luftdrucksensor - BMP085

Der Luftdrucksensor kann den Luftdruck in vier verschiedenen Genauigkeiten angeben. In der Standardauflösung werden zur Erhöhung der Genauigkeit intern zwei Samples nacheinander aufgenommen und das Ergebnis in 17 Bit abgelegt. Die Varianten *high resolution* (basierend auf vier Samplen) und *ultra high resolution* (acht Sample) belegen 18 bzw. 19 Bit – siehe Abbildung 6.32. Eine im Datenblatt als *ultra low power* bezeichneter Modus liefert dagegen einen 16 Bit breiten Wert. Außerdem liefert der Luftdrucksensor eine Temperatur, welche ebenfalls als 16-Bit-Wert zur Verfügung gestellt wird.

Die mögliche maximale Abtastrate ergibt sich aus der gewählten Auflösung, da das Samplen in den unterschiedlichen Genauigkeiten jeweils unterschiedliche Zeiten benötigt. So dauert das Abtasten der Temperatur mit jeweils 4.5 ms genau so lange wie das Abtasten des Luftdrucks bei geringster Auflösung (16 Bit). Die Abtastung in höherer Auflösung dauert 7.5 ms für 17 Bit, 13.5 ms für 18 Bit und 25.5 ms für 19 Bit. Die maximalen Abtastraten liegen also zwischen ∼222 Hz für 16 Bit und ∼39 Hz für 19 Bit, wobei diese hohen Abtastraten in der Praxis wenig Relevanz haben dürften, da sich Temperatur und Luftdruck im anvisierten Einsatzbereich nicht mit solchen Geschwindigkeiten ändern werden (siehe dazu auch Abschnitt 4.2.3).

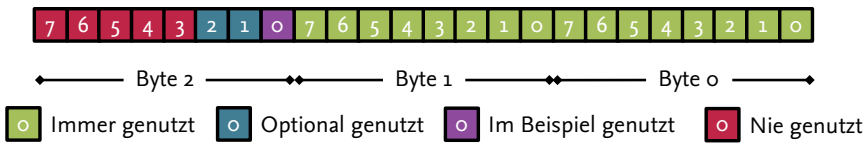


Abbildung 6.32: Drei Byte für die Ausgabe des Luftdrucks beim BMP085.

Zur besseren Ausnutzung der zur Übertragung zu verwendenden Bytes bietet sich auf den ersten Blick nur eine Zusammenfassung mehrerer zeitlich aufeinanderfolgender Werte an. Um kein Bit zu verschwenden, müssten bei einer Ausnutzung von 100 % 8 Messwerte in 19 Byte zusammengefasst werden.

Eine geschicktere Lösung ergibt sich, wenn man das Datenblatt bezüglich der Genauigkeit des Temperatursensors genauer liest – diese wird mit 0.1 °C angegeben. Der Messbereich geht von -40 bis 80 °C, was 1200 verschiedenen Werten entspricht, welche sich wiederum mit 11 Bit problemlos darstellen lassen. Das heißt von den vormals verwendeten 16 Bit lassen sich 5 Bit ohne einen realen Genauigkeitsverlust wegkürzen. Das wiederum bietet Spielraum, um die Daten von Luftdruck und Temperatur gemeinsam in weniger Bytes darzustellen. In Abbildung 6.33 ist zu sehen, dass (bei zwei ungenutzten Bits) Temperatur und Luftdruck in

nun 4 statt zuvor 5 Byte dargestellt werden können, was einer Reduktion auf 4/5, also auf 80.00 % der vorherigen Datenmenge, entspricht.

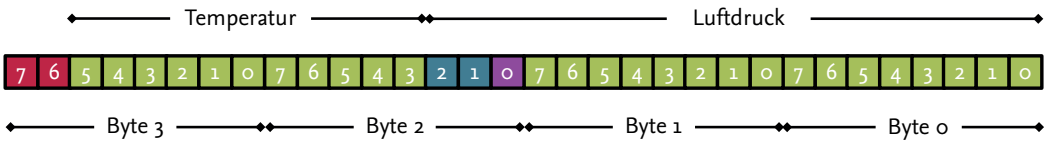


Abbildung 6.33: Fünf Byte für die Ausgabe von Luftdruck und Temperatur beim BMP085.

Interner AD-Wandler zur Strom- und Spannungsmessung

Die Überwachung der Batteriespannung und der aktuellen Stromaufnahmen erfolgen über den im Mikrocontroller integrierten ADC. Dieser bietet eine Auflösung von 10 Bit, welche standardmäßig in jeweils 2 Byte dargestellt werden. Wie in Abbildung 6.34 zu sehen, lassen sich diese 20 Bit bei vier ungenutzten Bits problemlos auch in 3 Byte darstellen, was dann 75.00 % (3/4) der vorherigen Datenmenge entspricht.

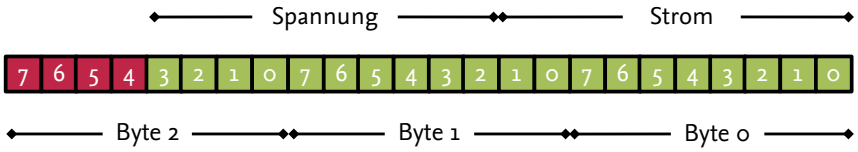


Abbildung 6.34: Drei Byte für die Messwerte von Strom und Spannung.

Zusammenfassung: Zustandslose Quellenkodierung

Durch ein einfaches Neuordnen der zu übertragenden Daten, lassen sich – je nach Sensor – bereits erhebliche Datenmengen sparen. In Tabelle 6.4 sind die möglichen Einsparungen für die auf INGA vorhandenen Sensoren zusammengefasst. Die *praktikablen* Werte basieren auf der gerade erwähnten Zusammenfassung zu *einem* Zeitpunkt und werden für die folgenden Messungen aus Gründen der Praktikabilität verwendet. Bei der Berechnung der *idealen* Werte, wurden auch mehrere zeitlich aufeinanderfolgende Messwerte zusammengefasst, um keine Bits ungenutzt zu lassen.

Tabelle 6.4: Einsparpotential durch zustandslose Quellenkodierung.

Sensor	Original-größe	Einsparung ideal	Einsparung praktikabel	Rest-redundanz	Neue Größe
Accelerometer ADXL345	3·2 Byte	18.75 %	16.67 %	1 Bit	5 Byte
Gyroskop L3G4200D	3·2 Byte	0 %	0 %	0 Bit	6 Byte
Luftdrucksensor BMP085	3+2 Byte	25 %	20 %	2 Bit	4 Byte
Strom- und Spannungsmessung	2+2 Byte	37.5 %	25 %	4 Bit	3 Byte

6.3.3 Analyse: Aufgenommene Daten aus realer Anwendung

Um einschätzen zu können, wie die auftretenden Daten nicht nur effektiv sondern auch effizient reduziert werden können, müssen zunächst die Eigenschaften der aufzuzeichnenden Daten bestimmt werden. Die in Abschnitt 6.3.1 kurz vorgestellte Huffman-Kodierung benötigt explizit die Auftrittswahrscheinlichkeiten der einzelnen Symbole um korrekt angewendet werden zu können.

Um eine Bewertungsgrundlage zu erhalten, wurden die Daten aller Sensoren von INGA aufgezeichnet, während INGA einen Tag lang am Körper getragen wurde. Die Versuchsdauer betrug dabei 24 Stunden, wobei die Sensoren wie folgt abgefragt wurden:

Accelerometer ADXL345 Abtastrate 100 Hz, Messbereich ±4 g, volle Auflösung (11 Bit), FIFO-Modus.

Gyroskop L3G4200D Abtastrate 100 Hz, Messbereich ±250 Grad pro Sekunde, volle Auflösung (16 Bit), FIFO-Modus.

Lufdrucksensor BMP085 Abtastung von Temperatur (16 Bit) und Luftdruck (19 Bit) im Wechsel mit einer Rate von 16 Hz.

Strom- und Spannungsmessung Referenzspannung zur Spannungsmessung $V_{R,U}$ =2.56 V, intern; Referenzspannung zur Strommessung $V_{R,I}$ =1.1 V, intern; Abtastung alle 50 ms (20 Hz).

Alle Daten wurde dabei zunächst unkomprimiert auf eine mircoSD-Karte geschrieben, wobei allerdings bereits die im vorherigen Abschnitt erläuterte zustandslose Quellenkodierung zum Einsatz kam. Für Accelerometer und Gyroskop wurden also über 8 Millionen Messwerte aufgezeichnet.

Accelerometer

Bei der in Abbildung 6.35 aufgetragenen Verteilung der Messwerte fällt zunächst auf, dass diese Daten nicht gleichverteilt sind, sondern mehrere Maxima aufweisen; außerdem treten viele Werte überhaupt nicht auf. Die Maxima lassen sich beispielsweise durch die Ruhelage (keine Beschleunigung, 0 g; bzw. Erdbeschleunigung 1 g) erklären. Der Messbereich umfasst ±4 g, welche mit 11 Bit aufgelöst werden, was einem Wertebereich von -1024 bis 1024 entspricht. 0 g entspricht demnach dem Wert „0“ und ±1 g den Werten ±256. Die nicht vorhandenen Messwerte am oberen und unteren Bereich der Skala resultieren daraus, dass der Messbereich nicht vollständig ausgenutzt wurden, also keine Beschleunigungen mit den vollen ±4 g aufgezeichnet wurden. Da jedoch auch vereinzelt Werte <-512 bzw >511 aufgetreten sind, kann der Messbereich nicht einfach von 11 Bit auf 10 Bit (±2 g) verkleinert werden, da sonst diese Werte verworfen würden.

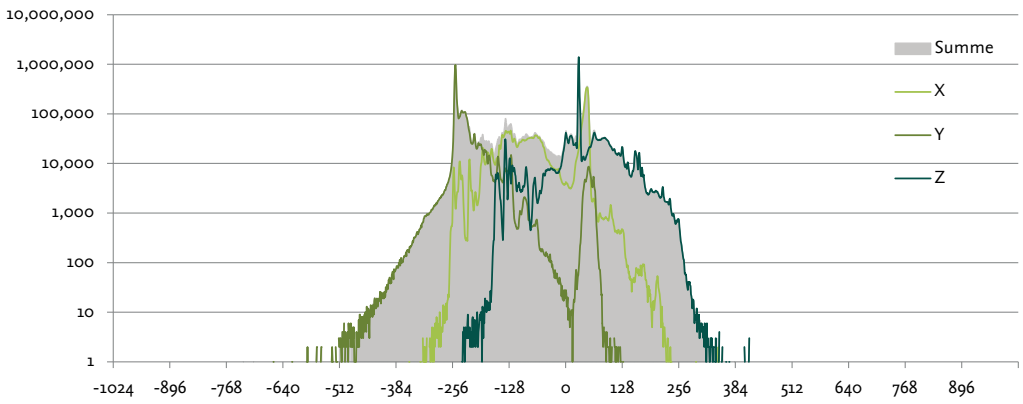


Abbildung 6.35: Verteilung der Messwerte des ADXL345 Accelerometers über 24 Stunden.

Gyroskop

In Abbildung 6.36 ist die Verteilung der Messwerte der drei Achsen des Gyroskops aufgetragen, wobei ein klares Maximum um den Wert 0 für alle Achsen zu erkennen ist. Auch ist zu erkennen, dass das gesamte Spektrum der zur Verfügung stehenden Werte nicht ausgeschöpft wird. Allerdings gilt hier dasselbe, das auch

schon für das Accelerometer galt: Würde man den Wertebereich halbieren (15 Bit statt 16 Bit), würden auch hier Messwerte verworfen werden, da vereinzelt Werte < -16 384 bzw > 16 383 auftreten.

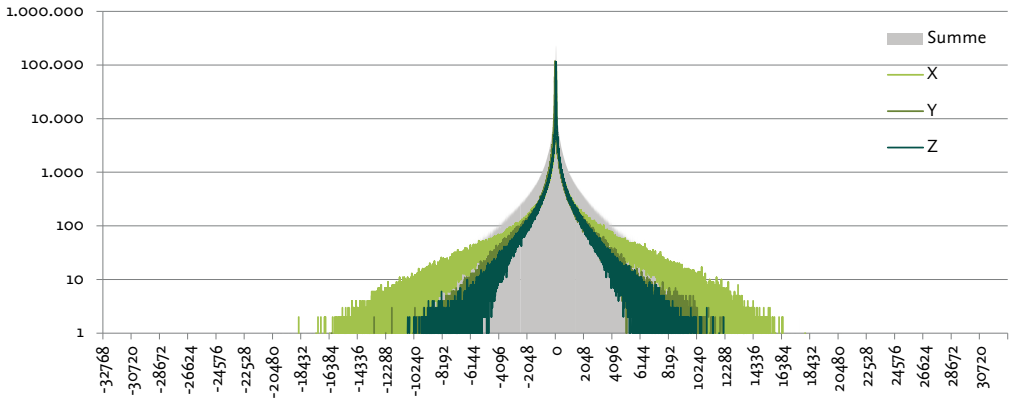


Abbildung 6.36: Verteilung der Messwerte des L3G4200D Gyroskops über 24 Stunden.

Luftdrucksensor

Die Verteilung der Werte des Luftdrucksensors ist in Abbildung 6.37 dargestellt. Da hier jedoch sowohl Luftdruck als auch Temperatur dargestellt sind und zwischen beiden große ungenutzte Bereiche liegen, lohnt an dieser Stelle eine Aufteilung in zwei unterschiedliche Wertebereiche.

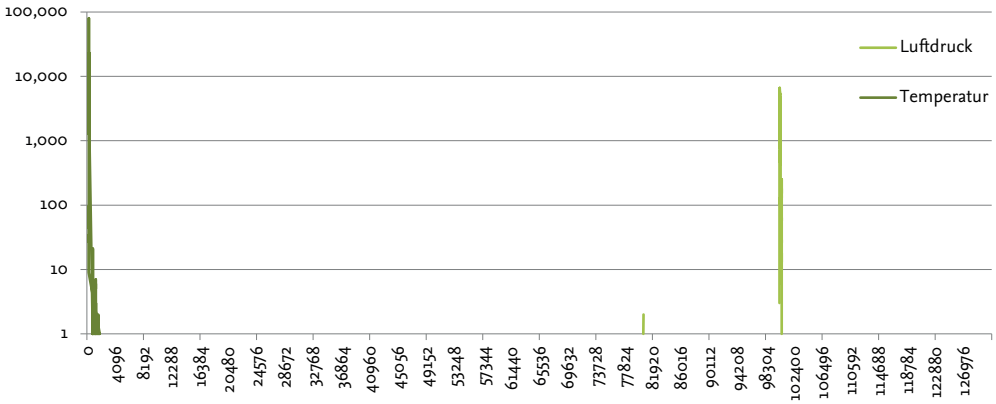


Abbildung 6.37: Verteilung der Messwerte für Temperatur und Luftdruck des BMP085 über 24 Stunden.

Der sehr beschränkte tatsächlich genutzte Wertebereich der exemplarischen Luftdruckmessung legt nahe, dass hier eine effiziente Kodierung mit nur wenigen Symbolen eine große Einsparung bringen kann. So zeigt Grafik 6.38, dass sich bei der exemplarischen Messung nahezu alle Werte im Intervall 100 318 bis 100 660 befinden, also von den zur Verfügung stehenden $2^{17} = 131\,072$ Werten gerade einmal 342 ($\ll 2^9$) überhaupt aufgetreten sind, sofern man die beiden Messwerte außerhalb des üblichen Messbereichs in Grafik 6.37 bei 80 625 einfach verwirft. Das legt nahe, diesen relevanten Bereich einfach zu maskieren und nur ihn bei der weiteren Datenverarbeitung zu berücksichtigen.

Da sich der absolute Luftdruck jedoch häufig ändert, würde sich dieser relevante Bereich, je nach herrschendem absoluten Luftdruck, verschieben – sei es durch eine veränderter Großwetterlage oder durch Einsatz in anderen Höhenregionen. Insofern eignet sich hier eine Maskierung ebenso wenig wie eine statische

Huffman-Kodierung.

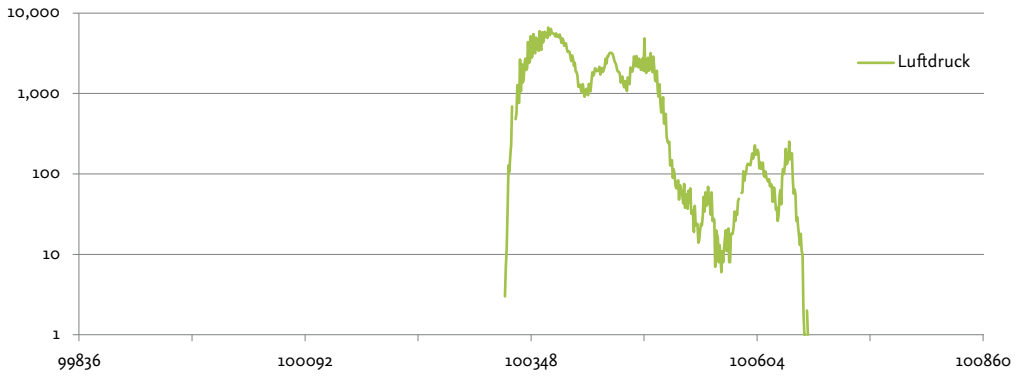


Abbildung 6.38: Verteilung der Messwerte für den Luftdruck des BMP085 über 24 Stunden.

Ganz ähnlich verhält es sich bei der gemessenen Temperatur; auch hier sind alle aufgenommenen Messwerte in einem sehr kleinen Intervall zwischen 256 und 384, was einem Dynamikumfang von 12.8 °C entspricht. Die Temperaturmessung wurde nicht kalibriert, weswegen über die tatsächlich herrschende Temperatur keine Aussage getroffen werden kann. Die Messungen wurden jedoch bei relativ mildem Klima im Sommer und einer Außentemperatur, die tagsüber nahe der Innentemperatur von 20 °C lag, durchgeführt. Bei anderen klimatischen Gegebenheiten (z.B. Tropen), ist eine Verschiebung dieses Bereichs zu erwarten; bei anderen Jahreszeiten (z.B. im Winter), ist hier ein größerer Dynamikumfang erwartbar.

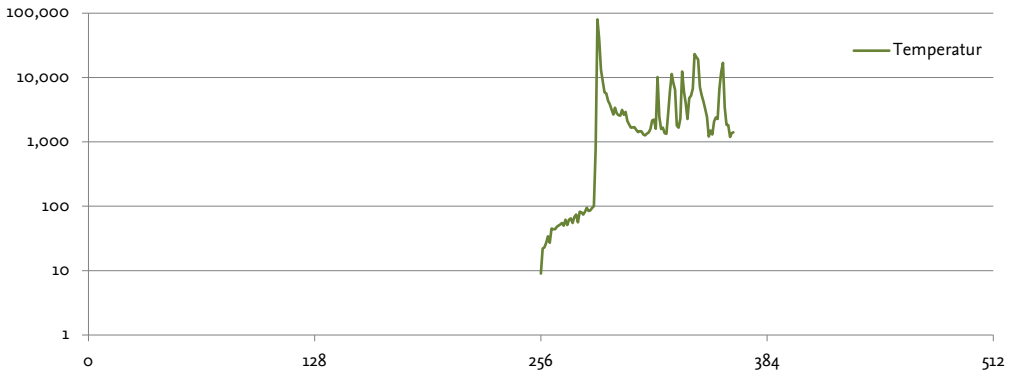


Abbildung 6.39: Verteilung der Messwerte für die Temperatur des BMP085 über 24 Stunden.

6.3.4 Strom- und Spannungsmessung

Die Strommessung findet immer zu einem diskreten Zeitpunkt statt, weswegen sie in ihrer Aussagekraft begrenzt ist. Wenn die Abtastzeiten der Strommessung beispielsweise phasenverschoben zur Aktivität bestimmter Stromverbraucher stattfinden, werden diese im schlechtesten Fall nicht erfasst.

Bei den in Abbildung 6.40 aufgetragenen Werten für die Strommessung fällt auf, dass diese ein globales Maximum und zwei lokale Maxima aufweisen. Da die Messungen zu diskreten Zeitpunkten stattfinden, kann davon ausgegangen werden, dass zu den jeweiligen Maxima jeweils unterschiedliche Systeme aktiv waren.

Messungen zur Stromaufnahmen der SD-Karte legen nahe, dass bei den höchsten Werten wahrscheinlich auf die SD-Karte geschrieben wurde, während die niedrigen Werte der Stromaufnahme im Leerlauf entsprechen.

Bei der Spannungsmessung ist die Aussagekraft größer, da hier in der Regel die Batteriespannung gemessen wird, welche mit der Zeit abnimmt, und auch keine hochfrequenten Sprünge zu erwarten sind. In Abbildung 6.40 ist auch die Verteilung der Messwerte für die Spannungsmessung mithilfe des internen AD-Wandlers aufgetragen. Es ist zu erkennen, dass sich im aufgezeichneten Fall, die Messwerte für die Spannung auf den Bereich zwischen 700 und 744 konzentrieren. Die beiden Widerstände R_5=1 MΩ und R_6=1.5 MΩ auf INGA (siehe Abschnitt 5.5.2) ergeben einen Spannungsteiler. Das heißt die tatsächliche Spannung U_{Bat} ergibt sich aus:

$$U_{Bat} = \frac{U_{AD} \cdot (R_5 + R_6)}{R_6} \tag{6.10}$$

U_{AD} wird dabei mit einer Auflösung von 10 Bit im Bereich zwischen 0 V und 2.56 V abgetastet. Das heißt, dass es in diesem Intervall 1024 linear verteilte Werte gibt, was wiederum 400 Werten pro Volt entspricht. Daraus ergibt sich für U_{AD}, dass eine Quantisierungsstufe 0.0025 V entspricht. Das Intervall zwischen 700 und 744 entspricht also tatsächlichen Batteriespannungen von 2.92 bis 3.10 V. Für die für den Versuch eingesetzten 2 Mignon-Zellen (AA) mit je 1.5 Volt ist das auch der erwartete Bereich. Bei anderen Spannungsquellen, würde sich dieser Bereich jedoch auch verschieben bzw. eine größere Dynamik aufweisen.

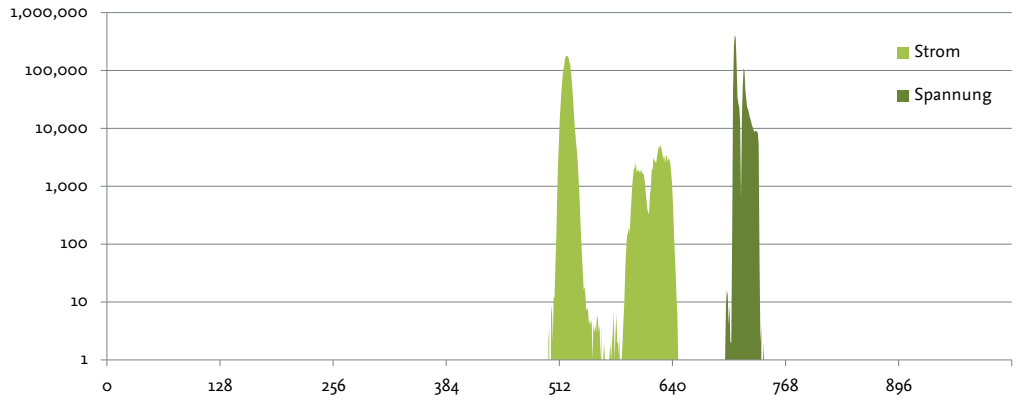


Abbildung 6.40: Verteilung der Messwerte für die interne AD-Wandlung bei der Strom- und Spannungsmessung.

6.3.5 Huffman-Kodierung der aufgenommenen Daten

Die grundsätzliche Funktionsweise der Huffman-Kodierung wurde bereits in Abschnitt 6.3.1 erläutert. Durch die Messungen im vorherigen Abschnitt lässt sich für die erwartbaren Daten ein Huffman-Baum entwerfen, der auf die konkreten Auftrittswahrscheinlichkeiten angepasst ist. Zur Berechnung des Huffman-Baums werden alle auftretenden Werte entsprechend ihrer Auftrittswahrscheinlichkeit kodiert. Auf einem 8-Bit-Mikrocontroller wird man eine solche Berechnung nicht ausführen wollen, da jeweils der komplette Wertebereich statistisch erfasst werden muss, um einen solchen Baum auch korrekt zu implementieren. Auf einem normalen PC lässt sich ein Baum solcher Größe jedoch problemlos berechnen.

Die Huffman-Kodierung ist – sofern sich die Auftrittswahrscheinlichkeiten nicht ändern – optimal. Das heißt, dass jedes Symbol im Mittel mindestens so viele binäre Stellen benötigt, wie sein Informationsgehalt – bzw. maximal eine zusätzliche. Die mittlere Wortlänge berechnet sich dabei aus der gewichteten Summe der

Wortlängen:

$$\bar{l} = \sum_{i \in X} p_i l_i \tag{6.11}$$

Da eine statische Huffman-Kodierung nur dann ideal ist, wenn alle Auftrittswahrscheinlichkeiten bekannt sind und sich im Verlauf nicht ändern, wird zunächst auf Basis des kompletten 24 h-Datensatzes ein optimaler Huffman-Baum für die erwarteten Daten generiert.

Huffman-Kodierung der Daten des Accelerometers

Innerhalb eines Tages sind beim Accelerometer insgesamt 50.226 MByte an Rohdaten angefallen. Diese Daten werden als 2 Byte große Werte vom Sensor zur Verfügung gestellt. In Abschnitt 6.3.2 wurde bereits gezeigt, wie diese Daten durch eine einfache Quellenkodierung um 16.6 % auf 41.855 MByte reduziert werden können.

Mit einem optimalen Huffman-Baum, welcher auf einem PC auf Basis der realen Verteilung der Messwerte errechnet wurde, ergibt sich im konkreten Fall eine mittlere Wortlänge vom 7.719 Bit. Die Daten lassen sich so auf 24.323 MByte reduzieren, was einer Reduzierung um 51.75 % entspricht. In Tabelle 6.5 sind die wichtigsten Parameter der Huffman-Kodierung des Accelerometers zusammengefasst. Die durchschnittliche Tripellänge (in Byte) gibt an, wie viele Bytes zum Kodieren eines Datensatzes (3 Achsen) benötigt werden. Zum Vergleich: Die Tripel der Rohdaten wurden mit 6 Byte und die Tripel der quellenkodierten Daten mit 5 Byte dargestellt.

Tabelle 6.5: Ideale Huffman-Kodierung des Accelerometers.

Rohdaten	Symbole	∅ Wortlänge	∅ Tripellänge	Daten	Einsparung
50.226 MByte	949	7.719 Bit	2.895 Byte	24.323 MByte	51.75 %

Um zu testen, in wieweit sich Abweichungen dieser Umsetzung auswirken, wird der Huffman-Baum in einem weiteren Versuch nur auf Basis des ersten Drittels der Daten (8 h) erzeugt und anschließend zum Kodieren der gesamten Daten verwendet. Es wird also ein nicht-idealer Huffman-Baum eingesetzt, was dazu führt, dass sich die mittlere Wortlänge auf 8.506 Bit verlängert. Das wiederum hat zur Folgen, dass die Daten nun zu insgesamt 26.670 MByte reduziert werden, was nur einer Reduzierung um 46.84 % entspricht.

Tabelle 6.6: Nicht-ideale Huffman-Kodierung des Accelerometers.

Rohdaten	Symbole	∅ Wortlänge	∅ Tripellänge	Daten	Einsparung
50.226 MByte	949	8.506 Bit	3.190 Byte	26.670 MByte	46.84 %

Für die Huffman-Kodierung der aufgenommenen Accelerometerdaten kommen insgesamt 949 unterschiedliche Symbole zum Einsatz. Wenn sich allerdings dieser Wertebereich verschiebt, verhält sich die Huffman-Kodierung alles andere als ideal. In Tabelle 6.7 wurde zur Verdeutlichung die durchschnittliche Wortlänge aufgetragen. Für alle Huffman-Kodierungen wird dabei derselbe Huffman-Baum angewendet. Bei der optimalen Kodierung sind die Daten entsprechend ihren Erwartungswerten verteilt. Der Wertebereich der aufgenommenen Daten wird nun jeweils um 20, 50, 250 und 500 Bit gegenüber der idealen Verteilung verschoben. Eine solche Verschiebung kann beim Accelerometer beispielsweise aus einer etwas anderen Ausrichtung am Körper resultieren: Wenn die Achsen nicht in die exakt selben Richtungen weisen, wie beim Generieren des Huffman-Baums, führt das zu einer Verschiebung des Wertebereichs, da sich damit die Auftrittswahrscheinlichkeit der einzelnen Symbole verändert.

Für die auswertenden Algorithmen ist eine solche Verschiebung in der Regel handhabbar, da entweder ohnehin nur relative Werte betrachtet werden, oder aber die Lage des Sensors am Körper rechnerisch kalibriert wird [150].

Zum Vergleich sind in Tabelle 6.7 auch die Wortlängen der unkodierten und quellenkodierten Daten aufgeführt. Es ist deutlich zu erkennen, dass jegliche Verschiebung des Wertebereichs bei der Huffman-Kodierung zu einer Verlängerung der durchschnittlichen Wortlänge führt. Das kann im schlechtesten Fall so weit gehen, dass bei einer Verschiebung des Wertebereichs um etwa die Hälfte, die Huffman-Kodierung eine längere durchschnittliche Wortlänge – und damit mehr Daten – produziert, als die mit viel Redundanz versehenen Rohdaten ohne Huffman-Kodierung benötigen.

Tabelle 6.7: Acclerometer: Durchschnittliche Wortlänge bei Verschiebung des Wertebereichs.

Aufgenommene Rohdaten	Quellen-kodierung	Huffman-Kodierung				
		optimal	+20	+50	+250	+500
16 Bit	13.33 Bit	7.72 Bit	8.79 Bit	10.04 Bit	15.03 Bit	19.53 Bit

Huffman-Kodierung der Daten des Gyroskops

Auch für das Gyroskop wurde ein idealer Huffman-Baum errechnet. In Tabelle 6.8 sind die wichtigsten Parameter dazu aufgelistet. Während die Daten des Gyroskops mit der einfachen Quellenkodierung nicht verringert werden konnten, ist nun die durchschnittliche Wortlänge mit 9.696 Bit deutlich geringer als die der Rohdaten (16 Bit), was sich auch in einer um 39.40 % verringerten Datenmenge niederschlägt.

Tabelle 6.8: Ideale Huffman-Kodierung des Gyroskops.

Rohdaten	Symbole	∅ Wortlänge	∅ Tripellänge	Daten	Einsparung
54.666 MByte	29 766	9.696 Bit	3.636 Byte	33.126 MByte	39.40 %

Wie in Tabelle 6.9 zu erkennen ist, wirkt sich auch beim Gyroskop eine Verschiebung des Wertebereichs negativ auf die durchschnittliche Wortlänge aus. Allerdings kann hier eine Verschiebung nicht ganz so einfach wie beim Accelerometer erreicht werden, da sich alle drei Achsen des Gyroskops ähnlich verhalten (siehe Abbildung 6.36. Wenn sich jedoch das Verhalten der Person ändert, die den Sensor trägt, oder aber der Huffman-Baum, der mit den Daten einer Person erzeugt wurde, dazu benutzt wird, die Daten einer anderen Person zu kodieren, kann auch beim Gyroskop von deutlich verschobenen Wertebereichen und damit einer nicht-idealen Huffman-Kodierung ausgegangen werden.

Tabelle 6.9: Gyroskop: Durchschnittliche Wortlänge bei Verschiebung des Wertebereichs.

Aufgenommene Rohdaten	Quellen-kodierung	Huffman-Kodierung				
		optimal	+20	+500	5000	+15000
16 Bit	16 Bit	9.70 Bit	11.47 Bit	14.01 Bit	18.90 Bit	24.80 Bit

Huffman-Kodierung der Daten des Luftdrucksensors

Mit einer Einsparung von 63.07 %, wie aus Tabelle 6.10 ersichtlich ist, lässt sich die Huffman-Kodierung besonders effizient auf den Luftdrucksensor anwenden. Von den ehemals 40 Bit (3 Byte für den Luftdruck und 2 Byte für die Temperatur) werden mit der Huffman-Kodierung durchschnittlich nur noch etwas weniger als 7.4 Bit benötigt.

Wie schon bei den beiden zuvor betrachteten Sensoren, wirkt sich auch hier eine Verschiebung des Wertebereichs sehr auf die durchschnittliche Wortlänge aus. In Tabelle 6.11 ist zu sehen, dass es auch hier – je nach Differenz – zu deutlichen Mehrkosten bei der Datenspeicherung und Übertragung kommen kann. Beim

Tabelle 6.10: Ideale Huffman-Kodierung für Luftdruck und Temperatur.

Rohdaten	Symbole	∅ Wortlänge	∅ Paarlänge	Daten	Einsparung
2.304 MByte	834	7.385 Bit	1.846 Byte	0.851 MByte	63.07 %

Einsatz in unterschiedlichen Temperaturbereichen oder bei sich (über die Zeit) veränderndem absoluten Luftdruck, werden solche Verschiebungen des Wertebereichs häufig auftreten.

Tabelle 6.11: Luftdrucksensor: Durchschnittliche Wortlänge bei Verschiebung des Wertebereichs.

Aufgenommene Rohdaten	Quellen-kodierung	Huffman-Kodierung				
		optimal	+20	+50	+100	+250
20 Bit	16 Bit	7.39 Bit	9.54 Bit	12.02 Bit	15.04 Bit	17.53 Bit

Huffman-Kodierung der Daten des internen AD-Wandlers

Die aufgenommenen Daten der Strom- und Spannungsmessung lassen sich ebenfalls sehr effizient kodieren. Wie in Tabelle 6.12 zu sehen, können die ursprünglichen 7.373 MByte Rohdaten um 68.52 % auf nur noch 2.321 MByte reduziert werden.

Tabelle 6.12: Ideale Huffman-Kodierung der Strom- und Spannungsmessung.

Rohdaten	Symbole	∅ Wortlänge	∅ Paarlänge	Daten	Einsparung
7.373 MByte	189	5.036 Bit	1.259 Byte	2.321 MByte	68.52 %

Der Kodierung kommt hierbei auch zugute, dass insgesamt überhaupt nur 189 verschiedene Symbole zum Einsatz kommen. Dass sich das bei einem verschobenen Wertebereich negativ auswirkt, zeigt Tabelle 6.13. Für die Stromaufnahme ist es allerdings bei gleichbleibendem Einsatzzwecke eher unwahrscheinlich, dass sich der Wertebereich verschiebt – es sei denn, eine veränderte Hardware oder ein anderes Scheduling [101] kommen zum Einsatz. Bei der Spannungsversorgung können unterschiedliche Arten von Batterien bzw. Akkumulatoren auch unterschiedliche Spannungsbereiche aufweisen, was wiederum in abweichenden Erwartungswerten für den Spannungsverlauf münden würde.

Zusammenfassung der Huffman-Kodierung

Zusammenfassend lässt sich für die Huffman-Kodierung sagen, dass sie durchaus in der Lage ist, das zu übertragende oder zu speichernde Datenaufkommen aller Sensoren signifikant zu reduzieren. In Tabelle 6.14 sind die Ergebnisse der idealen Huffmann-Kodierung für alle Sensoren zusammengefasst.

Beim Einsatz in der Praxis kann jedoch davon ausgegangen werden, dass ein idealer Huffman-Baum nicht erreicht werden kann, da anzunehmen ist, dass sich die Auftrittswahrscheinlichkeiten der einzelnen Symbole verändern. Schon eine geringe Abweichung – beispielsweise in der Ausrichtung des Accelerometers – hätte zur Folge, dass sich die Häufigkeitsverteilung der einzelnen Messwerte verschiebt und so der Baum nicht mehr zum gewünschten Kompressionsgrad führt. Im schlechtesten Fall könnte dann durch eine Huffman-Kodierung sogar eine größere Datenmenge als ohne Kodierung erzeugt werden.

Für eine effiziente Huffman-Kodierung sollte der zur Kodierung verwendete Huffman-Baum im Arbeitsspeicher des Mikrocontrollers abgelegt sein, damit auf die einzelnen Symbole und die entsprechenden Kodes auch schnell zugegriffen werden kann. Bei den 189 verschiedenen Symbolen für die Strom- und Spannungsmessung wären 2 Byte für den Messwert und 3 Byte für das kodierte Wort benötigt, da die Huffman-kodierten Worte bis zu 22 Bit lang sein können. Daraus folgt im konkreten Fall, dass der Huffman-Baum im Random-Access

Tabelle 6.13: Interner AD-Wandler: Durchschnittliche Wortlänge bei Verschiebung des Wertebereichs.

Aufgenommene Rohdaten	Quellen-kodierung	Huffman-Kodierung				
		optimal	+1	+5	+20	+100
16 Bit	12 Bit	5.04 Bit	5.30 Bit	6.50 Bit	13.50 Bit	15.70 Bit

Tabelle 6.14: Einsparpotential der idealen Huffman-Kodierung für alle Sensoren.

Sensor	Rohdaten	Symbole	∅ Wortlänge	Daten	Einsparung
Accelerometer ADXL345	50.23 MByte	949	8.51 Bit	26.67 MByte	46.84 %
Gyroskop L3G4200D	54.67 MByte	29 766	9.70 Bit	33.13 MByte	39.40 %
Luftdrucksensor BMP085	2.30 MByte	834	7.39 Bit	0.85 MByte	63.07 %
Strom- und Spannungsmessung	7.37 MByte	189	5.03 Bit	2.32 MByte	68.52 %

Memory (RAM) wenigstens 1.5 kByte belegt, was jedoch durchaus handhabbar ist. Die 949 verschiedenen Worte, die zur Huffman-Kodierung der Accelerometerdaten benötigt werden, können bis zu 25 Bit lang werden; sie benötigen also 4 Byte. Der gesamte Huffman-Baum für das Accelerometer würde demnach mindestens 5.5 kByte Arbeitsspeicher belegen; ähnlich viel Speicher würde auch für den Baum der Temperatur- und Luftdruckdaten benötigt. Ganz anders sieht es beim Gyroskop aus: Hier werden zwar auch maximal 25 Bit lange Werte kodiert, allerdings ist der Wertebereich der 16 Bit mit 29 766 verschiedenen Kodierungen zu einem Drittel ausgenutzt. Das führt dazu, dass der Huffman-Baum für das Gyroskop größer als 178.5 kByte wird. Diese Datenmenge kann vom verwendeten Mikrocontroller alleine im RAM nicht verwaltet werden; selbst bei Kombinationen der anderen Sensoren und deren Huffman-Bäumen, wird die Grenze des im Arbeitsspeicher Möglichen schnell erreicht. Es ist prinzipiell zwar auch möglich, einen Huffman-Baum im Flash-Speicher zu halten, was aber durch die benötigte Zugriffszeit (siehe Abschnitt 6.2.4) zu einem signifikanten weiteren Overhead und damit zu einer stark verringerten Abtastrate führt.

Eine weitere Einschränkung der Huffman-Kodierung für WBANs ergibt sich aus der Struktur der Daten: Ihren Gewinn zieht die Huffman-Kodierung aus den unterschiedlichen Symbollängen, d.h. häufig auftretende Werte führen zu kleineren Datenpaketen und weniger häufig auftretende Werte zu größeren Paketen. Die daraus resultierenden Pakete sind auf der einen Seite nicht so einfach weiterzuverarbeiten wie Pakete mit statischer Größe. Auf der anderen Seite kann dies auch ein Sicherheitsproblem darstellen: Wenn ein Angreifer die statistische Verteilung der Werte kennt, also ihm der verwendete Huffmann-Baum bekannt ist, kann er alleine aus der Länge der Daten auf deren Inhalt schließen. Dies ließe sich zwar durch Padding – also Auffüllen aller Pakete auf eine einheitliche Größe – bereinigen, allerdings würde auf diese Weise auch die erzielte Einsparung zunichte gemacht.

6.3.6 Deltakodierung

Am Anfang des Abschnitts wurde dargelegt, dass die Abtastrate sich an den zu erwarteten Frequenzen orientieren soll, um möglichst wenig Daten zu erzeugen. Da zum einen nicht immer von vornherein ersichtlich ist, welche Frequenzen im Maximum erreicht werden, zum anderen dieses Maximum auch eher die Ausnahme als die Regel darstellt, führt dies zu vielen ähnlichen Messwerten. So werden von einer kontinuierlichen Bewegung etliche Stützstellen aufgenommen – man spricht von einer Überabtastung. Messwerte zwischen zwei Punkten einfach zu verwerfen würde allerdings der Anforderung, alle Daten zu erhalten, widersprechen. Was bei der oft praktizierten Überabtastung jedoch zwangsläufig auftritt, ist eine zeitliche Korrelation der einzelnen Werte. Außerdem haben nahezu alle aufgenommenen Werte einen Bezug zu Bewegungen des menschlichen Körpers, an welchem die Sensoren ja für die Messungen befestigt waren. Erwartungsgemäß

führt das dazu, dass spontane und sehr große Änderungen der aufgenommenen Werte eher unwahrscheinlich sind, während keine oder nur geringe Änderungen die Regel darstellen.

Diese Eigenschaft macht sich die Deltakodierung zu Nutze: Wenn häufig auftretenden kleine Änderungen mit kleineren Symbolen kodiert werden können und größere (und seltenere) Änderungen dementsprechend mit längeren Symbolen kodiert werden, kann sich das nur positiv auf die zu übertragende Datenmenge auswirken. Das heißt, dass nach einem einmal aufgenommenen und komplett übertragenen Messwert nur noch die Differenz zum jeweils vorherigen Messwert übermittelt wird.

An einem einfachen Beispiel soll das an dieser Stelle verdeutlicht werden: In der Ruhelage ergeben drei aufeinanderfolgende Messungen der Beschleunigung dreimal denselben 11 Bit großen Messwert, also 33 Bit. Bei einer Deltacodierung müsste nur der erste Messwert (11 Bit) übertragen werden – für die beiden folgenden Messwerte würde lediglich ein Bit zur Übertragung der Änderung von „0“ benötigt, also 13 Bit insgesamt.

Auch für das HTTP gibt es beispielsweise eine Erweiterung zur Deltakodierung von Informationen, sodass bei leicht veränderten Daten, für die beim Client schon eine ältere Version zwischengespeichert ist, nur noch Differenzen übermittelt werden müssen [151].

Da so eine Abhängigkeit von zuvor aufgenommenen Daten erzeugt wird, führt der Verlust einzelner Daten auch zu einer Korruption der folgenden Daten. Diese und weitere Einschränkungen in der Praxis werden später in diesem Abschnitt erläutert.

Im Gegensatz zur Huffman-Kodierung, die im ungünstigsten Fall selten auftretende Werte auch mit Symbolen kodieren kann, die länger als die unkomprimierten Symbole sind, kann das bei der Deltakodierung nur im äußersten Extremfall auftreten, in dem der Wertebereich in zwei aufeinander folgenden Messungen einmal komplett durchschritten wird. Wenn beispielsweise in zwei aufeinanderfolgenden Messungen zuerst das positive Maximum und dann das negative Maximum gemessen wird, würde zur Deltakodierung dieses Sprungs ein weiteres Bit benötigt. Im konkreten Fall des Accelerometers würde das bedeuten, dass eine Änderung vom Maximum zum Minimum den aufgenommenen Wertebereich einer Differenz von -2047 und eine Änderung vom Minimum zum Maximum einer Differenz von +2047 entsprechen würde. Wären die Messwerte also unabhängig voneinander und gleichverteilt, würden damit für deltakodierte Messwerte des Accerelometers 12 Bit statt zuvor nur 11 Bit pro Achse benötigt.

Eine ideale Implementierung der Deltakodierung würde jeden folgenden Messwert mit genau der benötigten Anzahl an Bits kodieren und so eine maximale Ersparnis erreichen. In der Praxis wäre es dann aber sehr schwierig (bzw. unmöglich) zu erkennen, wann die Übermittlung des einen Wertes abgeschlossen ist und wann der nächste Wert folgt. Das anfängliche Beispiel krankt also daran, dass die Grenzen zwischen den einzelnen Messungen nicht klar erkennbar wären. Zwar lassen sich mit verschiedenen Konzepten auch Werte kodieren, deren Grenzen auch bei beliebiger Länge eindeutig erkennbar sind (beispielsweise SDNVs oder auch die gerade erwähnte Huffman-Kodierung), in der Praxis mit Mikrocontrollern ist das jedoch meist sehr ressourcenintensiv. Um schnell und effizient die Grenzen zwischen zwei Werten zu erkennen, wird daher auf Werte mit einer festgelegten Länge gesetzt, was auch im Folgenden berücksichtigt wird.

Nachdem nun die theoretischen Vor- und Nachteile der Deltakodierung erläutert wurden, soll zunächst bestimmt werden, inwiefern sich die praktisch aufgenommenen Daten zur Deltakodierung eignen.

Deltaverteilung der Accelerometerdaten

Um bestimmen zu können, in welchen Größenordnungen sich die einzelnen Daten von ihren jeweils zuvor aufgenommenen Daten unterscheiden, wurde aus den oben aufgetragenen Verteilungen die Deltaverteilung bestimmt. Die Daten wurden also auf die Änderung zum jeweils vorangegangenen Messwert hin untersucht. Dazu wird schlicht die Differenz des jeweils aktuellen Messwerts zum vorangegangenen Messwert bestimmt und diese dann gespeichert. Die Umsetzung ist also denkbar einfach und erfordert lediglich sehr wenig RAM, da nur der jeweils vorherige Messwert (im konkreten Fall 2 Byte) vorgehalten werden muss, um diese

Berechnungen auszuführen. Es ist jedoch äußerst wichtig, dass für jede gemessene Größe bzw. Achse eines Sensors ein Delta gebildet wird, da sonst die gewünschte Abhängigkeit der Messwerte verloren ginge; insofern erfolgt die Betrachtung für das Accelerometer pro Achse.

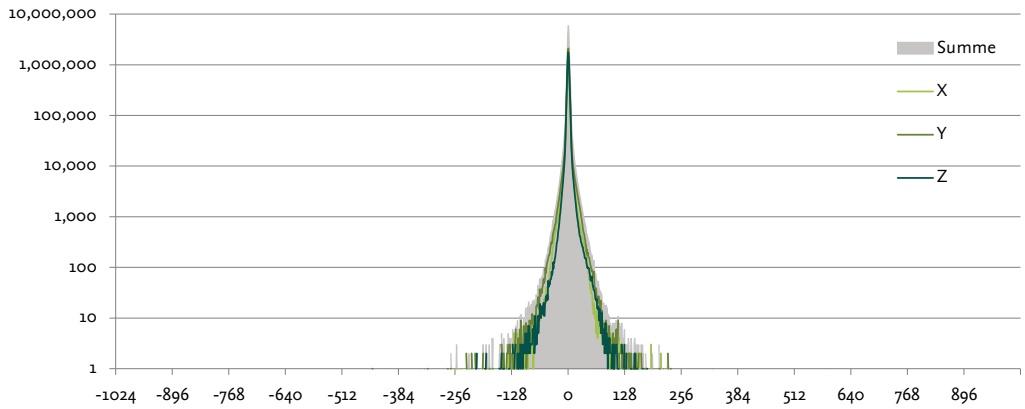


Abbildung 6.41: Auftrittshäufigkeit der Deltawerte zwischen zwei aufeinanderfolgenden Messwerten für die drei Achsen des Accelerometers.

In Abbildung 6.41 ist die Auftrittshäufigkeit der Änderungen zwischen zwei aufeinanderfolgenden Abtastungen der Accelerometerdaten aufgetragen. Es ist deutlich zu erkennen, dass das Maximum bei *keiner Änderung* liegt und sich der Großteil aller Werte in einem schmalen Bereich um den Wert „0“ häuft. Außerdem ist zu erkennen, dass der Bereich der X-Achse gar nicht erst den theoretisch möglichen Bereich von -2047 bis +2047 umfasst, da diese Änderungen in der Praxis nicht aufgetreten sind. Die logarithmische Darstellung der Werte in Abbildung 6.41 lässt vereinzelt auftretende Deltawerte im gesamten Bereich (beispielsweise bei -897 oder bei +869) außer Acht. In Tabelle 6.15 sind deshalb die Wertebereiche für mögliche Deltakodierungen aufgetragen. Es ist zu erkennen, dass der Wertebereich für eine Deltakodierung auch bei 11 Bit 100 % der aufgenommenen Werte umfasst. Bei einer Deltakodierung mit nur 6 Bit können immer noch ∼99.9 % aller Werte kodiert werden. Außerdem bemerkenswert ist, dass sich mehr als 42 % aller auftretenden Deltas mit nur einem Bit kodieren ließen.

Tabelle 6.15: Verteilung der Deltawerte des Accelerometers innerhalb kodierbarer Grenzen.

	Wertebereich		Werte im Wertebereich [%]			
Bit	von	bis	X-Achse	Y-Achse	Z-Achse	Summe
12	-2048	+2047	100.000 000	100.000 000	100.000 000	100.000 000
11	-1024	+1023	100.000 000	100.000 000	100.000 000	100.000 000
10	-512	+511	99.999 857	99.999 881	99.999 988	99.999 908
9	-256	+255	99.999 689	99.999 558	99.999 797	99.999 681
8	-128	+127	99.998 722	99.997 993	99.998 961	99.998 559
7	-64	+63	99.993 334	99.987 636	99.993 191	99.991 387
6	-32	+31	99.921 192	99.846 649	99.940 222	99.902 688
5	-16	+15	99.505 185	99.078 988	99.689 309	99.424 494
4	-8	+7	98.086 838	97.219 296	98.573 517	97.959 884
3	-4	+3	91.599 161	88.825 278	91.402 852	90.609 097
2	-2	+1	72.325 570	69.843 862	67.464 543	69.877 992
1	-1	0	45.038 716	43.555 323	39.088 848	42.560 962

Wollte man jedoch die vollen 100 % der aufgetretenen Messwerte verlustfrei mit einer Deltakodierung kodieren, wäre zunächst nichts gewonnen, da auch hier (genau wie bei der normalen Quellcodierung) 11 Bit pro Achse nötig wären.

Deltaverteilung der Gyroskopdaten

Auch für das Gyroskop wurden die Auftrittshäufigkeiten der einzelnen Abweichungen bestimmt. In Abbildung 6.42 ist eine Verteilung zu sehen, die der des Accelerometers (Abbildung 6.41) auf den ersten Blick ähnelt, jedoch eine größere Dynamik aufweist. So ist der Wertebereich der Deltas des Gyroskops um einiges größer und damit auch die Verteilung der Messwerte um den Nullwert um einiges breiter. Während beim Accelerometer deutlich über eine Million der Deltas beim Wert „0“ liegen, ist es beim Gyroskop in etwa eine Größenordnung weniger.

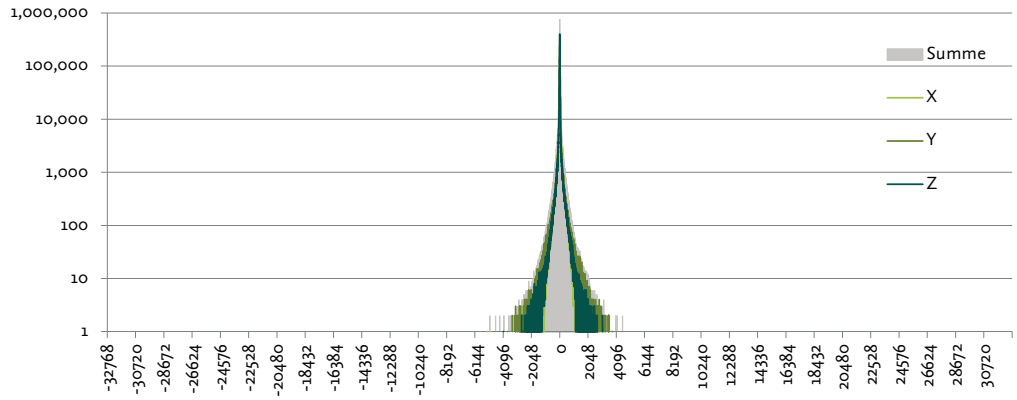


Abbildung 6.42: Auftrittshäufigkeit der Deltawerte zwischen zwei aufeinanderfolgenden Messwerten für die drei Achsen des Gyroskops.

In Tabelle 6.16 ist dann auch zu sehen, dass die vollen 100 % erst bei einer Kodierung mit 17 Bit erreicht werden. Etwas mehr als 99 % ließen sich mit 10 Bit kodieren und mit nur einem Bit würden gerade einmal 4.3 % der Deltas erfasst werden.

Eine reine Deltakodierung, die alle Messwerte des Gyroskops berücksichtigt, würde also mit 17 Bit sogar ein Bit mehr als die normale Quellcodierung benötigen. Die durchschnittliche Wortlänge beträgt hier 8.078 Bit.

Deltaverteilung der Daten des Luftdrucksensors

Mit dem Luftdrucksensor lassen sich sowohl Luftdruck als auch Temperatur in relativ hoher Auflösung abtasten. In Abbildung 6.43 ist die Verteilung dieser Deltas zu sehen. Auch hier ist eine sehr große Häufung um den Wert „0“ zu erkennen, welche besonders bei der Temperatur stark ausgeprägt ist. Allerdings weisen beide Messungen jedoch auch Ausreißer auf, weswegen auch hier jeweils der gesamte Wertebereich von 18 (Luftdruck) bzw. 12 Bit für eine hundertprozentige Abbildung benötigt würde.

In Tabelle 6.43 ist deshalb auch für diesen Sensor die prozentuale Verteilung der Deltawerte dargestellt. Während sich mehr als 97 % der Deltawerte für die Temperatur in nur einem Bit kodieren lassen, würden beim Luftdruck in diesem Fall nur etwas mehr als 16 % der Werte erfasst werden.

Deltaverteilung der Strom- und Spannungsmessung

Der interne ADC wird zur Strom- und Spannungsmessung verwendet. In Abbildung 6.44 sind die Auftrittshäufigkeiten der Deltas für diese Werte dargestellt. Es fällt auf, dass sich die Deltas der Messwerte für die Spannung wieder nahe der „0“ häufen, während die Deltas für die Stromstärke zwei weitere Maxima ausbilden.

Tabelle 6.16: Verteilung der Deltawerte des Gyroskops innerhalb kodierbarer Grenzen.

	Wertebereich		Werte im Wertebereich [%]			
Bit	von	bis	X-Achse	Y-Achse	Z-Achse	Summe
17	-65536	+65535	100.000 000	100.000 000	100.000 000	100.000 000
16	-32768	+32767	99.999 989	99.999 978	99.999 989	99.999 985
15	-16384	+16383	99.999 791	99.999 857	99.999 901	99.999 850
14	-8192	+8191	99.999 517	99.999 188	99.999 638	99.999 448
13	-4096	+4095	99.998 354	99.995 796	99.998 463	99.997 538
12	-2048	+2047	99.991 637	99.970 256	99.987 257	99.983 050
11	-1024	+1023	99.935 573	99.702 712	99.887 170	99.841 818
10	-512	+511	99.382 374	98.691 629	99.274 867	99.116 290
9	-256	+255	97.390 809	96.493 796	97.616 184	97.166 930
8	-128	+127	93.496 284	92.711 871	94.763 880	93.657 345
7	-64	+63	87.109 064	86.961 353	89.747 396	87.939 271
6	-32	+31	70.397 734	74.152 965	73.667 553	72.739 418
5	-16	+15	43.189 480	48.536 630	46.328 019	46.018 043
4	-8	+7	22.898 375	26.767 030	25.929 813	25.198 406
3	-4	+3	11.616 809	14.143 320	14.606 276	13.455 468
2	-2	+1	5.829 818	7.589 072	8.765 921	7.394 937
1	-1	0	2.920 902	4.278 960	5.827 941	4.342 601

Die allgemeine Verteilung der Messwerte wies schon ein ähnliches Bild auf. Daraus lässt sich nun schließen, dass die Stromaufnahme auch relativ häufig zwischen den in Abbildung 6.40 dargestellten Messwerten schwankt.

Die Abdeckung der einzelnen kodierbaren Deltabereiche ist wiederum in Tabelle 6.18 zu sehen. Während bei der Spannung mehr als 99.7 % der Werte bereits mit 2 Bit kodiert werden können, braucht es beim Strom 8 Bit, um diese Grenze zu überschreiten. Allerdings ist eine hundertprozentige Abdeckung der Messwerte für die Stromaufnahme bereits bei 10 Bit gegeben, während die Spannung erst bei einer Deltakodierung mit 11 Bit verlustfrei abgebildet wäre.

Zusammenfassung: Deltaverteilungen der Messwerte

Bei allen Sensoren sieht die Häufigkeitsverteilung der Deltawerte zunächst sehr vielversprechend aus. So lassen sich jeweils ein Großteil der aufgenommenen Deltas mit nur sehr wenigen Bit kodieren. Abbildung 6.45 zeigt dazu noch einmal die prozentuale Abdeckung deltakodierter Messwerte in Abhängigkeit der dazu verwendeten Anzahl an Bit.

Man könnte argumentieren, dass die Ausreißer in der Deltaverteilung höchstwahrscheinlich auf Mess- oder Verarbeitungsfehler hinweisen und sich jeweils auf einen bestimmten Prozentsatz festlegen, ab dem man gewillt ist, Werte als unrealistisch zu verwerfen und gar nicht erst zu kodieren. Dafür würde die Deltaverteilung der Temperaturwerte sprechen, da es bei am Körper getragenen Sensoren ziemlich unwahrscheinlich ist, dass sich zwischen zwei aufeinanderfolgenden Messwerten, die Temperatur um das volle Spektrum ändert; die mit 12 Bit darstellbaren Werte umfassen den Bereich größer ± 1024, was bei einer Auflösung von 0.1 °C pro Wert einer Schwankung von mehr als 100 °C entsprechen würde.

Würde man beispielsweise eine Abdeckung von mindestens 99 % als hinreichendes Ziel definieren, so würden für die Deltakodierung die in Tabelle 6.19 dargestellten Einsparungen erreicht werden. Die *idealen* Werte für die einzelnen Sensoren ergeben sich dabei aus den durch einzelne Bit kodierbaren Deltawerten.

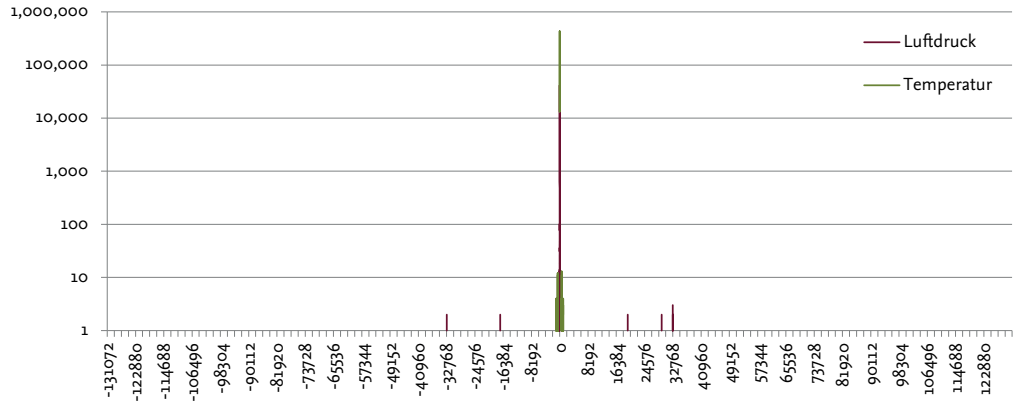


Abbildung 6.43: Auftrittshäufigkeit der Deltas zwischen zwei aufeinanderfolgenden Messwerten für Temperatur und Luftdruck.

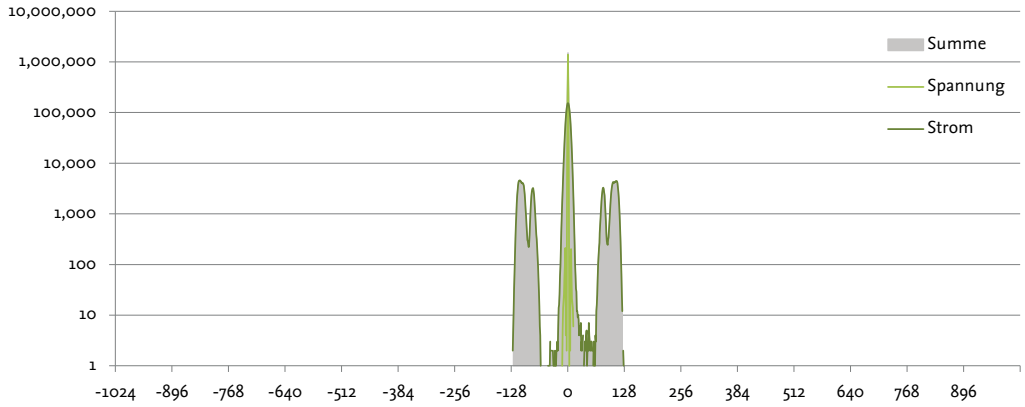


Abbildung 6.44: Auftrittshäufigkeit der Deltas zwischen zwei aufeinanderfolgenden Messwerten für Spannung und Stromstärke.

Praktikable Werte wiederum ergeben sich bei der Kodierung von ganzen Bytes, was für die Weiterverarbeitung sinnvoll ist.

Trotz beeindruckender Einsparmöglichkeiten von bis zu 75 % im betrachteten Beispiel birgt die alleinige Anwendung der Deltakodierung mehrere Fallstricke. Wenn (wie im Beispiel geschehen) nur 99 % der Werte berücksichtigt werden, könnten durch ein solches Vorgehen gerade die später interessanten Vorgänge nicht erfasst werden. Bei einem Sturz wird z.B. eine abrupte Änderung der Beschleunigung erwartet – und unter Umständen könnte diese Änderung so groß sein, dass sie gerade in die ausgeblendeten 1 % fällt. In sofern haben wir es wieder mit einer verlustbehafteten Kompression zu tun, welche von vornherein unerwünscht war. Wenn hingegen alle Werte bei einer Deltakodierung auch erfasst werden sollen, lässt sich – wie in Tabelle 6.20 ersichtlich – zwar bei manchen Sensoren theoretisch die Datenmenge marginal reduzieren, aber spätestens bei einer praktikablen Umsetzung und der Verwendung von ganzen Bytes wird jegliche Einsparung im Vergleich zur zustandslosen Quellenkodierung zunichte gemacht. Im Gegenteil, es werden bei den Daten des Gyroskops durch die Deltakodierung in diesem Fall sogar mehr Daten erzeugt.

Tabelle 6.17: Verteilung der Deltawerte für Luftdruck und Temperatur innerhalb kodierbarer Grenzen.

	Wertebereich		Werte im Wertebereich [%]		
Bit	von	bis	Luftdruck	Temperatur	Summe
18	-131072	+131071	100.000 000	100.000 000	100.000 000
17	-65536	+65535	99.991 536	100.000 000	99.995 768
16	-32768	+32767	99.964 627	100.000 000	99.982 313
15	-16384	+16383	99.882 378	100.000 000	99.941 189
14	-8192	+8191	99.867 404	100.000 000	99.933 702
13	-4096	+4095	99.867 404	100.000 000	99.933 702
12	-2048	+2047	99.867 404	100.000 000	99.933 702
11	-1024	+1023	99.867 404	99.985 460	99.926 432
10	-512	+511	99.867 404	99.932 292	99.899 848
9	-256	+255	99.867 404	99.917 535	99.892 469
8	-128	+127	99.866 970	99.917 318	99.892 144
7	-64	+63	99.861 545	99.917 318	99.889 431
6	-32	+31	99.764 322	99.917 318	99.840 820
5	-16	+15	98.799 043	99.917 318	99.358 180
4	-8	+7	86.503 877	99.917 318	93.210 597
3	-4	+3	56.739 489	99.917 318	78.328 403
2	-2	+1	30.643 730	99.917 318	65.280 524
1	-1	0	16.298 863	97.058 587	56.678 725

6.3.7 Angepasste Deltakodierung mit quellenkodierten Extremwerten

Wie gerade gezeigt wurde, ist eine alleinige Deltakodierung weder praktikabel noch zielführend. Zum einen werden bei der verlustfreien Kodierung teilweise mehr Daten erzeugt als ohne Deltakodierung; zum anderen wird durch die Deltakodierung eine Abhängigkeit zu vorherigen Messwerten erzeugt, die – im Falle eines fehlerbehafteten Übertragungskanals – zu einer endlosen Fehlerfortpflanzung führen kann. Bei einem Datenverlust auf der Funkstrecke sind auch die folgenden Daten korrumpiert.

Auch viele andere Kompressionsverfahren, die über verlustbehaftete Kanäle genutzt werden, nutzen die Differenz zu vorangegangenen Übertragungen, um Bandbreite zu sparen. In der Videoübertragung haben sich z.B. MPEG [152] und seine Nachfolger etabliert, wobei dort zunächst auch von einem Vollbild ausgegangen wird und anschließend die Differenzen zu diesem Vollbild (oder zu den vorher übermittelten Differenzen) übermittelt werden. Um Fehler zu kompensieren, wird bei MPEG in regelmäßigen Abständen ein Vollbild übermittelt, auf dessen Basis dann die weiteren Bilder berechnet werden können. Ein ähnliches Verfahren soll deshalb auch hier angewendet werden.

Des Weiteren haben die Untersuchungen im vorangegangenen Abschnitt gezeigt, dass bei der Deltakodierung der aufgenommenen Werte, sich jeweils ein Großteil dieser Deltawerte um den Nullpunkt häuft. So lassen sich beispielsweise bei der Temperatur über 99.9 % der Deltas mit nur zwei Bit kodieren. Den Wertebereich für die verbleibenden < 0.1 % von 2 auf 12 Bit zu erweitern, ist da wenig effizient.

Wenn also auf der einen Seite ohnehin periodisch quellenkodierte Messwerte übermittelt werden sollen, um fehlerhaften Übertragungen entgegenzuwirken, können auf der anderen Seite auch selten auftretende Extremwerte auf dieselbe Art übermittelt werden. Um jedoch die verschiedenen Datenformate unterscheiden zu können, bedarf es mindestens eines weiteren Bit, welches eine Unterscheidung zwischen quellenkodierten und deltakodierten Messwerten erlaubt. Es entstehen also zwei verschiedene Datenformate für die Messwerte:

Tabelle 6.18: Verteilung der Deltawerte für Spannung und Stromstärke innerhalb kodierbarer Grenzen.

Bit	Wertebereich		Werte im Wertebereich [%] (p_Δ)		
(b_Δ)	von	bis	Spannung	Strom	Summe
11	-1024	+1023	100.000 000	100.000 000	100.000 000
10	-512	+511	99.999 946	100.000 000	99.999 973
9	-256	+255	99.999 946	99.999 946	99.999 946
8	-128	+127	99.999 946	99.999 946	99.999 946
7	-64	+63	99.999 946	88.312 494	94.156 220
6	-32	+31	99.999 946	88.306 851	94.153 399
5	-16	+15	99.999 946	88.277 934	94.138 940
4	-8	+7	99.990 072	83.226 174	91.608 123
3	-4	+3	99.952 637	57.978 005	78.965 321
2	-2	+1	99.739 746	32.059 479	65.899 613
1	-1	0	86.485 561	16.444 399	51.464 980

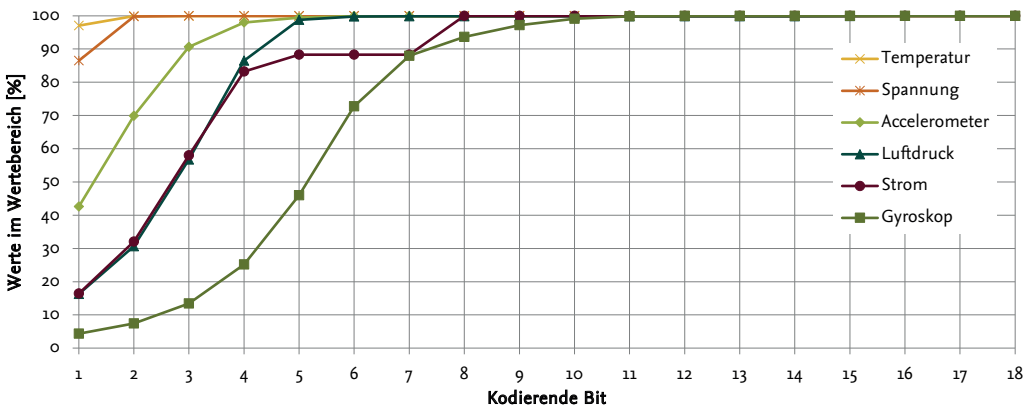


Abbildung 6.45: Prozentuale Abdeckung der einzelnen Messwerte bei Darstellung der Deltawerte variierender Bit-Anzahl.

Eines mit m+1 Bit für die quellenkodierten Daten und eins mit n+1 Bit für die deltakodierten Daten, wobei $m > n$, da sonst eine Deltacodierung keinen Sinn ergeben würde.

Eine *ideale* angepasste Deltakodierung für *einen* Messwert ergibt sich dabei jeweils aus der minimalen erzeugten Datenmenge. Die minimale Anzahl an Bit pro Messwert kann pro Deltaverteilung wie folgt bestimmt werden:

$$\min[p_{\Delta i} \cdot (b_{\Delta i} + 1) + (1 - p_{\Delta i})(b_{Sensor} + 1)] \tag{6.12}$$

Hierbei ist b_Δ die Anzahl der zur Deltakodierung genutzten Bit und p_Δ die Wahrscheinlichkeit, dass sich der Deltawert in diesem Bereich darstellen lässt. Lässt sich der Deltawert nicht darstellen, werden mit der Wahrscheinlichkeit $1 - p_\Delta$ die tatsächlich zur Abtastung genutzten Bit des Sensors b_{Sensor} verwendet. Bei beiden ist jeweils 1 Bit zu addieren, um beide unterschiedlichen Datenformate durch ein vorangestelltes Signalisierungsbit auseinanderhalten zu können.

Bei einer *realen* angepassten Deltakodierung ist vor allem die Darstellung in ganzen Bytes zu beachten, was teilweise eine deutliche Abweichung vom Ideal zur Folge hat. In Abschnitt 6.3.2 wurde bereits eine sinnvolle zustandslose Quellenkodierung vorgestellt, die bei der Deltakodierung nun jeweils als Rückgriff für Extremwerte, die nicht deltakodierbar sind, dienen soll. Die weitere Betrachtung sinnvoller Deltakodierungen innerhalb von Bytegrenzen finden für den jeweiligen Sensor im Folgenden statt.

Tabelle 6.19: Einsparpotential durch Deltakodierung bei 99 % Abdeckung.

Sensor	Quellen-kodiert	Deltakodierung ideal	Deltakodierung praktikabel	Einsparung ideal	Einsparung praktikabel	Neue Größe
Accelerometer ADXL345	5 Byte	3·5 Bit	16 Bit	62.50 %	60.00 %	2 Byte
Gyroskop L3G4200D	6 Byte	3·10 Bit	32 Bit	37.50 %	33.33 %	4 Byte
Luftdrucksensor BMP085	4 Byte	6+2 Bit	8 Bit	75.00 %	75.00 %	1 Byte
Strom- und Spannungsmessung	3 Byte	8+2 Bit	16 Bit	58.33 %	33.33 %	2 Byte

Tabelle 6.20: Einsparpotential durch Deltakodierung bei 100 % Abdeckung.

Sensor	Quellen-kodiert	Deltakodierung ideal	Deltakodierung praktikabel	Einsparung ideal	Einsparung praktikabel	Neue Größe
Accelerometer ADXL345	5 Byte	3·11 Bit	40 Bit	17.50 %	0.00 %	5 Byte
Gyroskop L3G4200D	6 Byte	3·17 Bit	56 Bit	-6.25 %	-16.67 %	7 Byte
Luftdrucksensor BMP085	4 Byte	18+12 Bit	32 Bit	6.25 %	0.00 %	4 Byte
Strom- und Spannungsmessung	3 Byte	10+11 Bit	24 Bit	12.50 %	0.00 %	3 Byte

Angepasste Deltacodierung für das Accelerometer

Aus den aufgenommenen Werten der Deltaverteilung des Accelerometers in Abbildung 6.41 lassen sich mehrere angepasste Deltakodierungen implementieren. Die quellenkodierten Messwerte werden dabei in jedem Fall mit $m = 11 + 1 = 12$ Bit kodiert. Für die Deltakodierung stehen nun pro Messwert 1 bis 10 Bit zur Verfügung, wobei jeweils auch das Signalisierungsbit zur Unterscheidung zwischen den beiden Kodierungen addiert werden muss. Es werden jeweils die Messwerte, die innerhalb der Bitgrenzen deltakodiert werden können, auch mit der Deltakodierung kodiert. Die Messwerte, die außerhalb des deltakodierbaren Bereichs liegen, werden quellenkodiert.

In Abbildung 6.46 sind für die Achsen des Accelerometers zunächst die erreichbaren Einsparungen gegenüber der zustandslosen Quellenkodierung aufgetragen. Die maximale Einsparung für die drei Achsen liegt jeweils bei 3 Bit für die Deltakodierung. Für die X-Achse ließe sich eine Einsparung von 45.38 % erzielen. In diesem Fall würden 91.60 % der Messwerte mit 3+1=4 Bit deltakodiert werden (siehe auch Tabelle 6.15) – die übrigen 8.40 % würden entsprechend mit 11 + 1 = 12 Bit quellenkodiert.

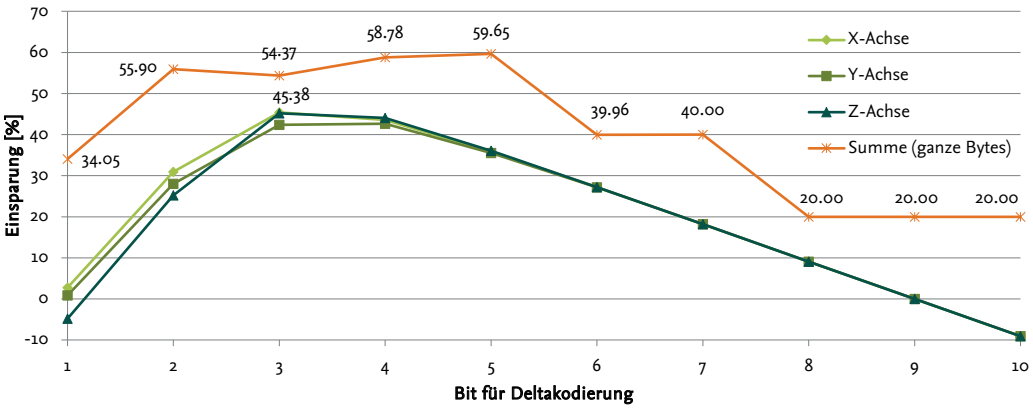


Abbildung 6.46: Datenreduktion der Accelerometerdaten durch die angepasste Deltakodierung.

Dieser Fall berücksichtigt jedoch keine Bytegrenzen, was eine Weiterverarbeitung erschwert. Bei der Übertragung von Tripeln (ein Messwert pro Achse) zur selben Zeit und der Berücksichtigung von Bytegrenzen, ergibt sich ein anderes Bild. Zunächst fällt auf, dass die Übertragung als Tripel in jedem Fall größere Einsparungen erlaubt, als die alleinige Übertragung der Werte je Achse. Das lässt sich dadurch erklären, dass in diesem Fall insgesamt weniger Signalisierungsbits zum Kennzeichnen der Daten benötigt werden. Außerdem lässt sich erkennen, dass das Maximum, welches die drei einzelnen Achsen jeweils an derselben Stelle aufweisen, nicht dem Maximum im kombinierten Fall entspricht. Bei einer Deltakodierung mit nur 3 Bit je Achse würde viel Redundanz geschaffen, da nur 10 Bit (3·3+1) in 2 Byte genutzt würden – die Gesamtersparnis läge bei lediglich 54.37 %. Das Optimum für diesen Fall liegt demnach bei einer Deltakodierung mit 5 Bit pro Achse – bei einer so erzielten Einsparung von 59.65 %. Mit 3·5+1=16 Bit werden so auch 2 Byte ohne verbleibende Redundanz ausgenutzt. Die Struktur der quellencodierten Daten entspricht der in Abbildung 6.31; das Datenformat der deltacodierten Daten ist in Abbildung 6.47 zu sehen.

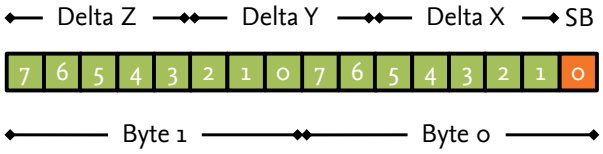


Abbildung 6.47: 2 Byte für die mit jeweils 5 Bit deltakodierten drei Achsen des Accelerometers.

Die angepasste Deltakodierung für das Accelerometer benötigt im konkreten Fall im Mittel 5.379 Bit pro Symbol. Das bedeutet, dass sie eine größere Einsparung ergibt, als die eigentlich optimale Huffman-Kodierung, die für das Accelerometer auf denselben Daten eine mittlere Wortlänge von 7.719 Bit aufweist. Erklären lässt sich dies mit dem unterschiedlichen Informationsgehalt: Während es bei der Huffman-Kodierung möglich ist, aus jedem übertragenen Symbol, den entsprechenden unkodierten Wert mit Hilfe des Huffman-Baums zu errechnen, benötigt die Deltakodierung zusätzlich den jeweils vorangegangenen Wert.

Angepasste Deltakodierung für das Gyroskop

Auch für die aufgenommenen Messwerte des Gyroskops lassen sich mehrere Deltakodierungen implementieren. In diesem Fall werden die quellenkodierten Messwerte in jedem Fall mit 16 + 1 = 17 Bit kodiert. Für die Deltakodierung stehen dementsprechend nun pro Messwert 1 bis 16 Bit zur Verfügung, wobei diese wiederum um ein Signalisierungsbit ergänzt werden müssen.

In Abbildung 6.46 sind nun für die Achsen des Gyroskops die erreichbaren Einsparungen gegenüber der zustandslosen Quellenkodierung aufgetragen, wobei hier bei einer kleinen Anzahl an Bit sogar eine Verschlechterung erzielt werden kann. Die maximale Einsparung für die drei Achsen liegt in diesem Fall jeweils bei 7 Bit für die Deltakodierung. Bei der X-Achse ließe sich hier eine Einsparung von 44.23 % erzielen, wobei dann 87.11 % der Messwerte mit 7+1=8 Bit deltakodiert werden (siehe auch Tabelle 6.16) – die übrigen 12.89 % würden entsprechend mit 16 + 1 = 17 Bit quellenkodiert.

Eine tatsächliche Quellenkodierung mit 17 Bit pro Achse wäre aber alles andere als wünschenswert, da so 7/8 des dritten Bytes nicht genutzt würden, was zu einer signifikanten Verschlechterung der erzielbaren Einsparung in der realen Anwendung führen würde. In Abschnitt 6.3.2 wurde bereits aufgezeigt, dass die drei 16 Bit-Messwerte der Gyroskopachsen sich zwar ohne verbleibende Redundanz in 6 Byte kodieren lassen, aber auch keine Einsparung bei „geschickterer“ Quellenkodierung erlauben. Eine Erweiterung dieser 6 Byte um ein weiteres Bit zur Unterscheidung zwischen delta- und quellenkodierten Daten wäre also unvorteilhaft und es ließe sich so nur eine maximale Einsparung von 41.96 % erzielen.

Um dennoch beide Datenformate unterstützen zu können, wird in diesem Fall ein weiteres Byte mit

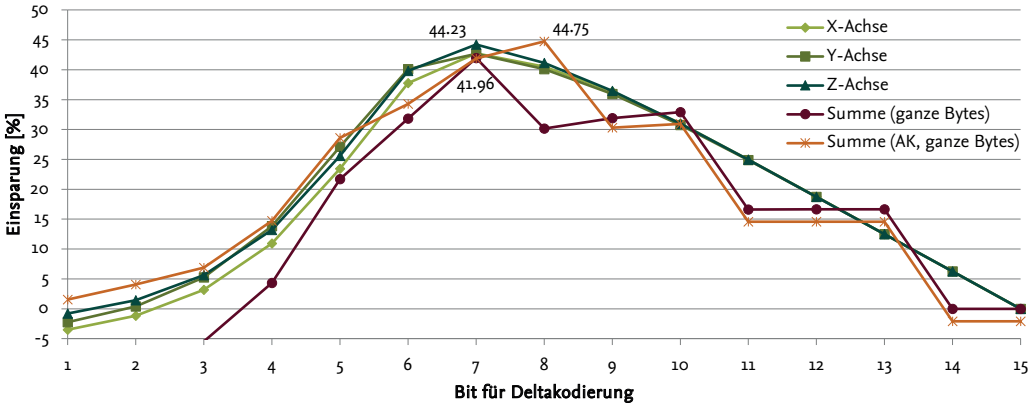


Abbildung 6.48: Datenreduktion der Gyroskopdaten durch die angepasste Deltakodierung.

den Siganlisierungsbits für die folgenden acht übertragenen Datensätze eingefügt. Mit dieser alternativen Kodierung (AC) lässt sich eine maximale Einsparung von 44.75 % bei einer Deltakodierung von jeweils 8 Bit pro Achse zu erreichen.

Abbildung 6.49 zeigt auf der rechten Seite das Byte, welches die Signalisierungsbits für acht aufeinanderfolgende Datensätze enthält; darauf folgt ein deltakodierter Datensatz.

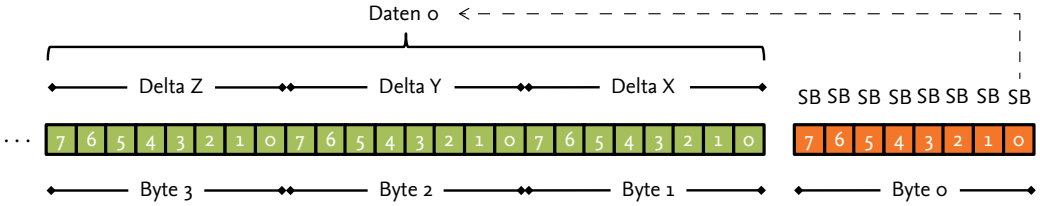


Abbildung 6.49: 3 Byte für die mit jeweils 8 Bit deltakodierten drei Achsen des Gyroskops; rechts davon das Byte mit den Signalisierungsbits für 8 Datensätze.

Angepasste Deltacodierung für den Luftdrucksensor

Die unterschiedliche Beschaffenheit der Messwerte dieses Sensors führt zu einer gewissen Besonderheit. Wie in Abbildung 6.50 zu sehen ist, weisen die Kurven der Deltawerte für Luftdruck und Temperatur kein gemeinsames Maximum auf. Vielmehr sinkt die Kurve für die Temperatur stetig, beginnend ab einer Einsparung von 68.98 % bei einer idealen angepassten Deltakodierung mit nur 1 Bit. Die Kurve für den Luftdruck besitzt bei 5 Bit ein Maximum bei einer Einsparung von 58.05 %. Würde man beide unterschiedlichen Deltawerte mit derselben Anzahl an Bit kodieren, läge das Maximum bei 3 Bit je Messwert und einer Einsparung von lediglich 58.75 %. In diesem Fall würden also zwei 3 Bit-Werte um ein Signalisierungsbit erweitert und zu einem Byte mit einer Restredundanz von 1 Bit zusammengefasst.

Um eine höhere Einsparung zu erreichen, ist es im konkreten Fall daher ratsam, die einzelnen Deltas unterschiedlich zu kodieren. Das einfache Addieren der beiden Maxima würde zu einer 5 Bit-Deltakodierung für den Luftdruck und einer 1 Bit-Deltakodierung der Temperatur führen. Da – unter Beirücksichtigung des zusätzlich benötigten Signalisierungsbits – so jedoch nur 7 von 8 möglichen Bit eines Bytes ausgenutzt werden, ergäbe sich eine Gesamtersparnis von 71.92 %. Das tatsächliche Maximum der Einsparung bei der Kombination beider Deltawerte liegt jedoch bei 74.041 %, die erreicht werden, wenn die Temperatur mit 2 und der Luftdruck mit 5 Bit deltakodiert werden – so lässt sich ein Byte komplett ausnutzen.

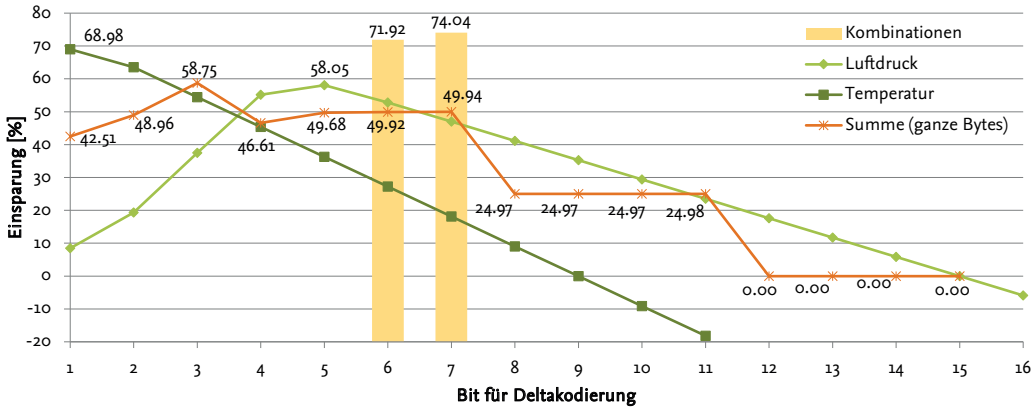


Abbildung 6.50: Datenreduktion der Gyroskopdaten durch die angepasste Deltakodierung.

Während die Quellcodierung der Werte des Luftdrucksensors schon in Abbildung 6.33 zu sehen war, ist in Abbildung 6.51 das zur Deltakodierung benötigte Byte abgebildet.

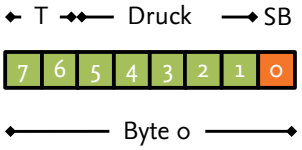


Abbildung 6.51: Mit 2 Bit kodierte Temperaturdifferenzen und mit 5 Bit kodierte Druckdeltas ergeben ein vollständiges Byte.

Angepasste Deltacodierung für Stromstärke und Spannung

Die Deltas der Messwerte des internen AD-Wandlers weisen ebenfalls eine unterschiedliche Charakteristik auf, wie sie auch schon in Abbildung 6.44 erkennbar ist. Während die Werte für die Spannung sich um den Wert „0“ häufen, weist die Strommessung größere Deltas mit zwei weiteren Maxima auf.

In Abbildung 6.52 bestätigt sich dieses Bild. Während die alleinige und ideale Deltakodierung der Werte für die Spannung ihr Maximum bei 2 Bit mit einer Einsparung von 59.66 % aufweist, liegt das Maximum der deltakodierten Stromstärke bei 4 Bit bei einer Ersparnis von lediglich 21.55 %. Durch die relativ breite Streuung dieser Werte ist eine Deltakodierung überhaupt erst ab 4 Bit sinnvoll, da bei weniger eine Verschlechterung im Vergleich zur reinen Quellenkodierung eintritt.

Würde man beide Messgrößen mit derselben Anzahl von Bit deltakodieren, hätte das eine Ersparnis von 52.64 % zur Folge, wobei in diesem Fall die Deltas für Strom und Spannung jeweils mit 3 Bit kodiert wären und ein Bit ungenutzt bliebe. Kombiniert man aber beide Maxima und kodiert somit beide Messwerte unterschiedlich – also 2 Bit für die Deltas der Spannungsmessung und 5 Bit für die Deltas der Strommessung – ergibt sich eine Gesamtersparnis von 58.70 %. Mit einem weiteren Bit zur Signalisierung der unterschiedlichen Datenformate ist auch ein komplettes Byte gefüllt und es ergibt sich eine Repräsentation, die der in Abbildung 6.51 für den Luftdrucksensor entspricht. Für den Fall, dass die Spannung mit nur einem Bit kodiert wird, würde wiederum ein Bit ungenutzt bleiben, was in einer Einsparung von nur 50.90 % münden würde.

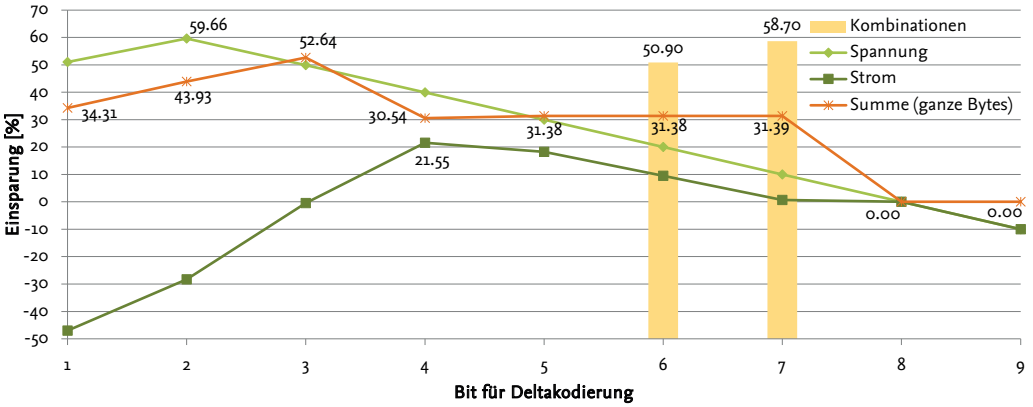


Abbildung 6.52: Datenreduktion der Messwerte für Stromstärke und Spannung durch die angepasste Deltakodierung.

Einfügen von quellenkodierten Messwerten

Die gerade vorgestellten Verfahren der angepassten Deltakodierung mit einer Quellenkodierung der Extremwerte basiert auf statistischen Annahmen. Werte, die sich nicht in den gewählten Grenzen deltakodieren lassen, werden jeweils in der zustandslosen Quellenkodierung kodiert. In Tabelle 6.21 sind zusammenfassend noch einmal die genutzten Deltakodierungen der einzelnen Sensoren dargestellt.

Tabelle 6.21: Messwertverteilung der angepassten Deltakodierungen für die einzelnen Sensoren.

	Delta-kodierung		Abgedeckter Wertebereich	Einsparung zu Q.-Kodierung	Gesamt
Accelerometer ADXL345	3·5 Bit	2 Byte	99.424 494 %	59.65 %	66.38 %
Gyroskop L3G4200D	3·8 Bit	3 Byte	93.657 345 %	44.75 %	44.75 %
Luftdrucksensor BMP085	5+2 Bit	1 Byte	98.717 353 %	74.04 %	79.23 %
Strom- und Spannungsmessung	5+2 Bit	1 Byte	88.048 187 %	58.70 %	69.02 %

Dabei ist allerdings zu beachten, dass der abgedeckte Wertebereich zwar etwas über die Auftrittswahrscheinlichkeit der einzelnen Deltas aussagt, jedoch nichts darüber, wann der durch Deltakodierung abgedeckte Wertebereich verlassen wird. So können – beispielsweise in Phasen mit nur geringer Bewegung – durchaus sehr lange Zeiträume mit der Deltakodierung kodiert werden; solange keine Extremwerte auftreten, würden auch keine quellenkodierten Daten kodiert und versendet. Bei einem fehlerbehafteten Kanal würde sich ein eventuell aufgetretener Fehler also so lange fortpflanzen, bis der deltakodierbare Wertebereich verlassen wird.

Um diese Fehlerfortpflanzung einzudämmen, können in periodischen Abständen quellenkodierte Datensätze eingefügt werden. Je nach Sensor verringert sich dabei die Gesamtersparnis geringfügig – je nach Periode P_{QK} der zur Sicherung eingefügten quellencodierten Datensätze. Der Faktor für den Mehraufwand für die Fehlertoleranz Q_{FT} ergibt sich dabei aus dem Größenverhältnis der quellenkodierten Datensätze B_{QK} zu den deltakodierten Datensätzen B_Δ.

$$Q_{FT} = \frac{B_{QK} + (P_{QK} - 1)B_\Delta}{P_{QK} \cdot B_\Delta} = \frac{B_{QK}}{P_{QK} \cdot B_\Delta} + \frac{(P_{QK} - 1)}{P_{QK}} \tag{6.13}$$

Im Fall des Accelerometers ist B_{QK} =5 Byte und B_Δ =2 Byte. Eine Periode P_{QK} = 1 bedeutet, dass jeder Wert quellenkodiert wird - in diesem Fall steigt das Datenaufkommen um den Faktor $Q_{FT} = 2{,}5$ – was dem Verhältnis zwischen B_{QK} und B_Δ entspricht. Wenn jeder zehnte Wert quellenkodiert werden soll (P_{QK} = 10),

würde im konkreten Fall $Q_{FT} = 1{,}15$ sein.

In Abbildung 6.53 ist zu sehen, wie sich das Einfügen von quellencodierten Datensätzen zur Vermeidung der Fehlerfortpflanzung auswirkt. Ab jeder Periode $P_{QK} > 1$ wird dabei generell eine Einsparung erreicht. Bei größer werdender Periode ist für jeden Sensor eine starke Annäherung an das zuvor erreichte Optimum zu beobachten, wobei 50 % des Optimums schon ab $P_{QK} = 3$ erreicht wird. 80 % werden für alles Sensoren ab $P_{QK} = 7$, 90 % ab $\sim P_{QK} = 15$ und 99 % ab $\sim P_{QK} = 150$ erreicht.

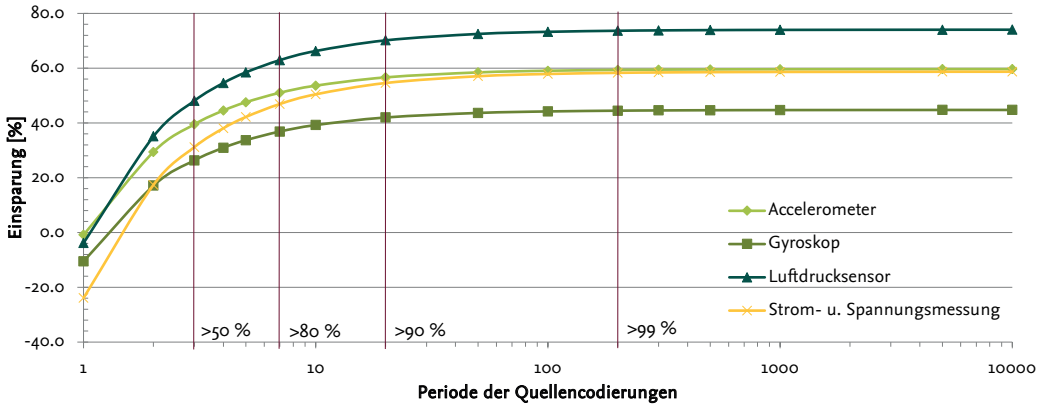


Abbildung 6.53: Erreichbare Einsparung bei unterschiedlichen Perioden für die Quellencodierung.

6.3.8 Zusammenfassung der Datenreduktion

Es gibt durchaus verschiedene Strategien, die anfallenden Rohdaten zu reduzieren – aus den gegebenen Anforderungen wurde sich auf Möglichkeiten zur verlustfreien Reduktion der Daten beschränkt. Dazu wurden die zustandslose Quellenkodierung, die Huffman-Kodierung und die Deltakodierung auf ihre Möglichkeiten und Schwächen hin untersucht. Die einzelnen Einsparpotentiale für die konkreten Sensoren bei dem betrachteten Fall der am Körper aufgenommenen Daten, sind in Tabelle 6.22 zusammenfassend dargestellt, wobei die 99 %-Deltakodierung verlustbehaftet ist.

Tabelle 6.22: Praktikables Einsparpotential aller evaluierten Methoden zur Datenreduktion.

	Quell-	Huffman-	Deltakodierung		
Sensor	kodierung	Kodierung	99 %	100 %	angepasst
Accelerometer ADXL345	16.67 %	46.84 %	60.00 %	0.00 %	66.38 %
Gyroskop L3G4200D	0.00 %	39.40 %	33.33 %	-16.67 %	44.75 %
Luftdrucksensor BMP085	25.00 %	63.07 %	75.00 %	0.00 %	79.23 %
Strom- und Spannungsmessung	37.50 %	68.52 %	33.33 %	0.00 %	69.02 %

Dabei ist die zustandslose Quellenkodierung sicherlich der einfachste Weg, um durch eine geschickte und an den Anwendungsfall angepasste Anpassung der zu übertragenden Bytes eine signifikante Reduktion des zu speichernden und/oder zu übertragenden Datenaufkommen zu erreichen. Lediglich beim Gyroskop, welches die 2 Byte langen Ausgaben pro Achse tatsächlich ausnutzt, lässt sich so keine Einsparung erzielen.

Die Huffman-Kodierung funktioniert zwar theoretisch bezüglich des Informationsgehalts optimal und benötigt zur Dekodierung neben dem Vorhandensein des Huffman-Baums keine weiteren Informationen, jedoch ist sie im skizzierten Anwendungsfall nur sehr schwierig umzusetzen: Bei einer Verschiebung des Wertebereichs ist die Reduzierung der Daten nicht mehr optimal – es können im schlechtesten Fall sogar

größere Datenmengen als ohne statische Huffman-Kodierung erzeugt werden. Außerdem belegen die zur Kodierung benötigten großen Bäume eine nicht zu vernachlässigende Menge an Arbeitsspeicher. Bei sehr kleinen Bäumen und einem weit geringerem Wertebereich, ließe sich zwar auch eine dynamische Huffman-Kodierung implementieren, für den Anwendungsfall ist das aber aufgrund der Größe des Wertebereichs schlicht nicht durchführbar.

Die für den Anwendungsfall sinnvollste und effizienteste Kodierung ist die angepasste Deltakodierung, die in ihrem Einsparpotential sogar über dem der eigentlich optimalen Huffman-Kodierung liegt und sich außerdem sehr einfach berechnen lässt. Die angepasste Deltakodierung macht sich dabei zu Nutze, dass die aufgenommenen Werte sehr stark voneinander abhängen und spontane Änderungen in großen Schritten eher selten auftreten. Eine Abhängigkeit von der Lage oder dem Einsatzort wie bei der Huffman-Kodierung gibt es so nicht, da lediglich eine Änderung der Dynamik sich auf das Auftreten der unterschiedlichen Deltawerte auswirken würde. Diese Effizienz wird dabei allerdings durch eine Abhängigkeit der einzelnen Messwerte zu ihren jeweiligen Vorgängern „erkauft“. In der realen Anwendung, in der trotz Verwendung von unterbrechungstoleranten Übertragungsprotokollen von einem unvorhersehbaren zwischenzeitlichen Datenverlust ausgegangen werden muss, sollte allerdings sichergestellt werden, dass in bestimmten Abständen auch rein quellenkodierte Datensätze übermittelt werden, um eine Fehlerfortpflanzung zu verhindern.

7 Mobiler Notfallkanal

„What happens on the road, stays on the road“

Sprichwort

Mit der bisherigen Umsetzung können innerhalb der Funkreichweite der Basisstation die im aggregierten Szenario definierten primären Anforderungen erfüllt werden. Auch außerhalb der Wohnung funktioniert dank der in Abschnitt 6.1 eingeführten DTN-Technik die nahtlose Aufzeichnung von Daten für eine spätere Auswertung. Wie in Abschnitt 2.3.2 erläutert, ist eine Outdoor-Funktionalität des Gesamtsystems – also auch eine Sturzerkennung und eine Notfallmeldung außerhalb der eigenen vier Wände – zumindest wünschenswert. Damit wäre auch dem in Abschnitt 2.2.2 erläuterten *Vermeidungsverhalten* begegnet [41].

In diesem Kapitel wird deshalb ein automatisches und autonomes System entworfen, implementiert und evaluiert, welches es ermöglicht, Stürze auch außerhalb der Wohnung zu detektieren und entsprechende Alarmmeldungen über einen mobilen Notfallkanal abzusetzen.

Zur Erkennung eines Sturzes innerhalb der Wohnung wurden in Kapitel 2 bereits einige Systeme kurz vorgestellt. Für eine Sturzerkennung außerhalb der Wohnung eignen sich jedoch nur tragbare Sensoren, welche bereits in Abschnitt 2.1.1 aufgeführt wurden.

7.1 Sturzerkennung auf mobilen Geräten

Bisher wurde davon ausgegangen, dass die Algorithmen zur Sturzerkennung auf der leistungsfähigen Hardware der Basisstation innerhalb der Wohnung laufen. Wenn sich die stürzende Person aber außerhalb der Funkreichweite befindet, kann dieser Sturz also im bisherigen Design nicht erkannt werden. Es gilt also zunächst, die Sturzerkennung auch auf mobilen Geräten zu implementieren.

Heutige Smartphones sind mittlerweile so leistungsfähig wie frühere Supercomputer [153]. Dass eine mobile Alarmmeldung und Sturzerkennung möglich ist, zeigen zahlreiche Anwendungen für Smartphones. Applikationen wie *PerFallD* [25] und *iFall* [26] benutzen den Beschleunigungssensor eines Android-Smartphones zur Sturzerkennung. Außerdem nutzen die Applikationen das Mobilfunknetz, um Notfallmeldungen abzusetzen; so wird quasi ein vollständiges mobiles Hausnotruf- und Sturzerkennungssystem implementiert.

PerFallD liefert als einzige bekannte Publikation auch Aussagen zur Batterielaufzeit. Die Autoren schreiben von einer Laufzeit von 33.5 Stunden auf einem *HTC Dream* bzw. *T-Mobile G1* Smartphone. Im Datenblatt des betreffenden Gerätes schreibt der Hersteller von einer Standby-Zeit von ∼402 Stunden – diese (zumeist allerdings fiktive) Lebensdauer wurde damit also um mehr als den Faktor 10 verringert. Wenn man zusätzlich davon ausgeht, dass bei beiden Angaben jeweils peinlich genau darauf geachtet wurde, dass keinerlei andere Applikationen laufen und somit Strom verbrauchen – das Smartphone also nicht als solches genutzt wird – wird schnell klar, dass die Batterielebensdauer unter normalen Nutzungsbedingungen wohl eher weit unter diesen Angaben liegen wird. In Szenarien, in dem ein Smartphone tatsächlich und häufig genutzt wird, hält der Akkumulator nie länger als wenige Stunden [154] – auch ohne zusätzliche Sturzerkennung.

7.1.1 Einschränkungen von Smartphones

Während also die Rechenleistung moderner Smartphones dem Gesetz von Moore [155] folgt, gilt das aber nicht für die Kapazität von Batterien oder Akkumulatoren. Ein bei Smartphonebesitzern (die ihr Gerät

Tabelle 7.1: Sturzerkennung mit unterschiedlichen Systemen (aus [41])

	Feste Installation[1]	Smartphone	tragbares Sensorsystem
Rechenkapazität	sehr hoch	hoch	stark eingeschränkt
Speicherkapazität	sehr hoch	hoch	stark eingeschränkt
Energiebeschränkungen	keine[2]	sehr hoch	normal
Verfügbare Sensoren	alle	wenige	einige[3]

auch regelmäßig für diverse Tätigkeiten nutzen) häufig zu beobachtendes Verhalten ist, dass das Gerät zwar unterwegs in einer Tasche am Körper getragen wird, sobald aber der Arbeitsplatz oder die Wohnung erreicht ist, es zum Aufladen an ein Ladegerät angeschlossen wird. Der hohe Energiebedarf ist also ein Faktor, der dazu führt, dass das Smartphone unter Umständen gerade dann nicht am Körper getragen wird, wenn es gebraucht wird, sondern in diesem Moment an der Steckdose lädt. Auch herkömmliche Mobiltelefone werden von manchen Menschen nur außerhalb der Wohnung am Körper getragen, während innerhalb der Wohnung die Nutzung eines Festnetztelefons das bevorzugte Kommunikationsmittel darstellt.

Ein ganz anderer Punkt bei der Verwendung von Smartphones zur Sturzerkennung ist die unbestimmte und unter Umständen wechselnde Position der Sensoren. Manche Menschen tragen das Gerät in der Hosentasche, andere am Gürtel, wieder andere in der Innentasche der Jacke oder gar in einer separaten Handtasche. Zwar kann die Ausrichtung des Geräts mit entsprechenden Algorithmen korrigiert werden [150], für eine zuverlässige Sturzerkennung ist aber auch ein Wissen um die Position und die Art der Befestigung des Sensors (ob eher lose oder eher fest) unerlässlich [20].

Wenn es also eine Lösung geben soll, in der ein einzelnes Gerät sowohl in der Wohnung als auch außerhalb der Wohnung eine Sturzerkennung und Alarmmeldung durchführen soll, so kann das nicht das Smartphone sein. Auf der anderen Seite ist es wahrscheinlich auch nicht ratsam, zwei parallele Systeme – eines für drinnen und eines für draußen – zu betreiben, da dadurch wohl eher Verwirrung und Unsicherheit geschaffen würde.

7.1.2 Vor- und Nachteile des tragbaren Sensorsystems

Wie schon in Abschnitt 2.1.1 erläutert, werden häufig mit Accelerometern ausgestattete Sensorknoten zur Sturzerkennung genutzt. Im Gegensatz zum Smartphone erlaubt der Einsatz von dedizierten Sensorknoten eine relativ genaue und immer ähnliche Positionierung dieser am Körper der zu überwachenden Person: Sei es durch die Befestigung am Gürtel oder gar die Integration in Schmuck- oder Kleidungsstücke. Des Weiteren ist ein Sensorknoten wie INGA mit einem festgelegten und immer gleichem Set an Sensoren ausgestattet, während die im Smartphone verbauten Sensoren von Hersteller zu Hersteller und Typ zu Typ variieren. Während für die im Smartphone verbaute Sensorik in der Regel nur Abstraktionen auf höheren Ebenen bereitstehen, die – abhängig von Betriebssystem und Nutzerrechten – einen eingeschränkten Zugriff auf die Rohdaten des einzelnen Sensors erlauben, können beim Einsatz von Sensorknoten auch bestimmte Interruptfunktionen der Sensoren genutzt werden und so energie- und recheneffizientere Algorithmen eingesetzt werden.

Auf der anderen Seite ist ein Sensorknoten in jedem Fall ein zusätzliches Gerät, da er alleine normalerweise nicht in der Lage ist, Alarmmeldungen abzusetzen, sobald er sich außerhalb der Funkreichweite der Basisstation befindet. Da ein Sensorknoten also auch nicht als alleiniges Gerät für eine automatische und autonome Sturzerkennung und Alarmmeldung infrage kommt, gilt es im Folgenden, die Vorteile beider Ansätze zu kombinieren und so eine nachhaltige und praktikable Lösung zu schaffen. In Tabelle 7.1 sind die bisher diskutierten Systeme gegenübergestellt.

7.2 Systemdesign zur Unterstützung mobiler Alarmmeldungen

Als Konsequenz aus der Tatsache, dass es kein einzelnes System zur Sturzerkennung und Notfallmeldung geben kann, welches drinnen wie draußen funktioniert und dabei keine Einbußen gegenüber vorhandenen Systemen für den Wohnungsbereich aufweist, wird eine Kombination aus Sensorknoten und Smartphone propagiert. Dieses System besteht aus drei Teilen und ist in Abbildung 7.1 skizziert. Dabei nimmt ein und derselbe Sensorknoten die Messwerte zur Sturzerkennung auf, während unterschiedliche Systeme für das Versenden von Notfallmeldungen zuständig sind: Innerhalb der Wohnung ist die Basisstation (A in Abbildung 7.1) dafür zuständig – genauso wie in den Szenarien, die in den vorigen Kapiteln diskutiert wurden. Sobald die Funkreichweite (d) der Basisstation verlassen wird, übernimmt ein mitgeführtes Smartphone oder Mobiltelefon diese Aufgabe (B in Abbildung 7.1). Im ersten Fall werden die Notfallmeldungen über das Internet oder eine normale Telefonleitung abgesetzt; im zweiten Fall wird dazu das Mobilfunknetz (2/3/4G) genutzt. Die vom Mobiltelefon abgesetzten Nachrichten können dabei auch um Positionsdaten (aus Netzwerkparametern oder GPS-Daten) erweitert werden und so den Rettungskräften auch den (ungefähren) Ort des Notfalls mitteilen.

In diesem Szenario können die Algorithmen zur Sturzerkennung entweder dem Sensorknoten oder auf einem Smartphone implementiert werden.

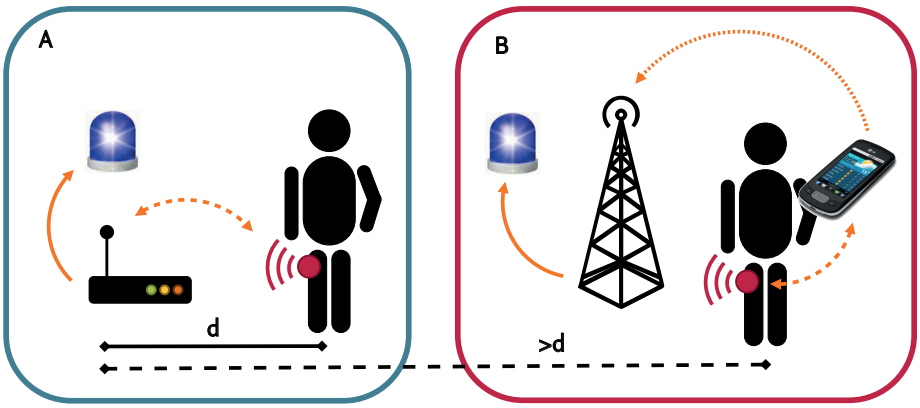


Abbildung 7.1: Alarmmeldung innerhalb der Funkreichweite d der Basisstation über die Basisstation (A) und außerhalb der Funkreichweite ($> d$) über das Mobilfunknetz mit Hilfe eines Smartphones (B).

7.2.1 Anbindung des Smartphones

Als drahtloser Sensorknoten verfügt INGA über ein IEEE 802.15.4-Funkinterface, um mit anderen Sensorknoten oder der Basisstation zu kommunizieren. Smartphones und Mobiltelefone verfügen jedoch in der Regel nicht über ein solches Funkinterface. Die Funkschnittstelle, die in diesem Bereich am häufigsten anzutreffen ist, ist Bluetooth (IEEE 802.15.1). Da sich der quelloffene Sensorknoten einfacher modifizieren lässt als jedes Mobiltelefon, wurde INGA um ein Bluetooth-Funkinterface erweitert.

Abbildung 7.2 zeigt eine solche Erweiterung und wie sie auf INGA (Version 1.2) montiert ist. Mit dem dort auch zu erkennenden blauen Kabel (auf INGA) wurde eine Interrupt-Leitung des Accelerometers mit einem Port-Pin des Mikrocontrollers verbunden, um die Sturzerkennung auch energiesparend realisieren zu können; in späteren Versionen von INGA ist diese Modifikation nicht mehr notwendig. Der dort verbaute

[1] Mit den in Abschnitt 1.1.4 kurz angesprochenen Sensoren.
[2] Ohne tragbare Sensorik.
[3] Abhängig von der Ausstattung des tragbaren Sensorsystems.

Bluetooth-Adapter kommuniziert über eine UART-Verbindung mit einer UART-Bridge, die an INGAs MSPI-Bus angeschlossen ist.

Um INGA unter Contiki mit einem Smartphone über Bluetooth zu koppeln, wurden entsprechende Treiber implementiert. Außerdem wurde ein Algorithmus zur Sturzerkennung auf INGA implementiert, der sich auch die Interrupts des Accelerometers zunutze macht [156]. In der umgesetzten Implementierung funktioniert INGAs Taster auch als Notfalltaster, der manuell auch dann eine Alarmmeldung initiieren kann, wenn automatisch kein Sturz erkannt wurde.

Ein Android-basiertes Smartphone stellt in diesem Fall die Gegenstelle dar. Hierfür wurden drei verschiedene Applikationen umgesetzt, die alle das Mobilfunknetz zum Absetzen der Notfallnachrichten nutzen:

- Im ersten Fall werden die Rohdaten des Beschleunigungssensors des Sensorknotens kontinuierlich über die Bluetoothverbindung an das Smartphone übermittelt und dort ausgewertet. Dazu wurde auf dem Smartphone ein Sturzerkennungs- und Kalibrierungsalgorithmus [150] in Java implementiert.

- In einem zweiten Szenario findet die gerade beschriebene Sturzerkennung nur auf dem Sensorknoten statt. Dort detektierte Stürze werden als Nachrichten über die Bluetooth-Verbindung zum Smartphone gesendet, welches dann über das Mobilfunknetz (je nach Konfiguration) einen Anruf tätigt oder eine Kurznachricht versendet.

- Zum Vergleich wurde auch eine alleinige Sturzerkennung auf dem Smartphone mit denselben Algorithmen implementiert, welche in diesem Fall die Messwerte des im Smartphone verbauten Accelerometers nutzt.

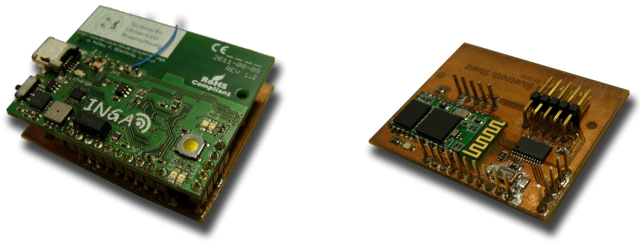


Abbildung 7.2: Links: INGA-Sensorknoten mit darunterliegendem Bluetooth-Adapter zur Kommunikation mit Mobiltelefonen; rechts: Bluetooth-Adapter ohne INGA.

7.3 Evaluation der Umsetzungen

Zur Evaluation wurde das gerade beschriebene System umgesetzt. Ein Motorola Milestone (Android 2.3.7, Cyanogen Mod 7.1.5) dient dabei als Smartphone; INGA mit Bluetooth-Erweiterung als Sensorknoten. Eine qualitative Aussage zur Leistungsfähigkeit der einzelnen Algorithmen zur Sturzerkennung kann an dieser Stelle nicht geleistet werden. Das Ziel dieser Evaluation ist zum einen, die grundsätzliche Machbarkeit zu zeigen; zum anderen soll bestimmt werden, welcher Teil des Systems am besten geeignet ist, die Algorithmen zur Sturzerkennung auszuführen.

Da eingangs behauptet wurde, dass der Energieverbrauch bei Smartphones ein stark limitierender Faktor ist, soll hier zunächst der Energieverbrauch des Smartphones in allen drei Implementierungen bestimmt werden. Anschließend erfolgt eine ähnliche Analyse für den Sensorknoten INGA.

7.3.1 Baseline: Energieverbrauch des Smartphones

Wie schon in Abschnitt 7.1.1 kurz angerissen, ist es schwer, eine allgemeingültige Aussage zum Energieverbrauch und damit zur Lebensdauer von Smartphones zu treffen. Zu viele Faktoren, wie das Nutzungsverhalten oder die Netzabdeckung, beeinflussen den Stromverbrauch. Der Hersteller des verwendeten Smartphones gibt für das Modell eine Standby-Zeit von 8 Stunden im UMTS-Modus an. Wenn diese Zeit überhaupt erreicht werden kann, dann nur ohne jegliche Benutzung. In den folgenden Untersuchungen wurden keine Telefongespräche geführt und alle unnötigen Systemprozesse und Applikationen wurden entfernt oder deaktiviert. Die Batteriespannung wurde von einer eigenen Anwendung kontinuierlich überwacht. Die „Idle"-Kurve in Abbildung 7.3 zeigt das Ergebnis dieser Messung.

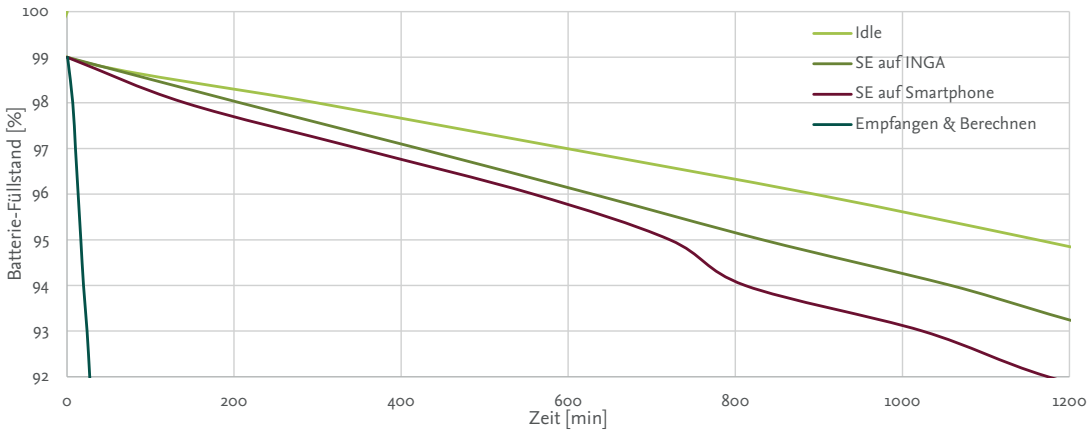


Abbildung 7.3: Batteriefüllstand des Smartphones bei den unterschiedlichen Implementierungen der Sturzerkennung.

7.3.2 Messwerte vom Sensorknoten – Sturzerkennung auf Smartphone

Die Datenaufnahme mit einem Sensorknoten hat den Vorteil, dass dieser Daten von einer definierten Position am Körper liefern kann. Außerdem ist der Sensor entsprechend den Anforderungen der auswertenden Algorithmen komplett konfigurierbar; er kann also Daten in der gewünschten oder benötigten Auflösung und Abtastrate liefern. Diese Daten werden im Indoor-Szenario zur Basisstation gesendet und können dort mit leistungsfähiger Hard- und Software ausgewertet werden.

Für das Outdoor-Szenario kann ein Smartphone die Basisstation ersetzen; da MSHP und Smartphone beide JAVA unterstützen, kann hierbei sogar derselbe Code verwendet werden. Die Daten mit dem Sensorknoten aufzunehmen und auf einem leistungsfähigen Smartphone auszuwerten, bietet für eine mobile Sturzerkennung also ähnliche Voraussetzungen wie im stationären Fall.

So einfach und naheliegend diese Umsetzung auch ist, so wenig energieeffizient ist sie leider auch. Die „Empfangen und Berechnen"-Kurve in Abbildung 7.3 zeigt den Verlauf der Batteriespannung des Smartphones, wenn dieses einzelne Messwerte vom Sensorknoten über Bluetooth empfängt und verarbeitet. Zur Verdeutlichung sind die einzelnen „Empfangen und Berechnen"-Kurven in Abbildung 7.4 in einem kleineren Ausschnitt aufgetragen: Während die anderen Szenarien im oberen Bereich zu verharren scheinen, wird durch das Empfangen und Verarbeiten der einzelnen Samples (nahezu unabhängig von der Datenrate) die Batterie des Smartphone innerhalb von weniger als 5 Stunden komplett geleert.

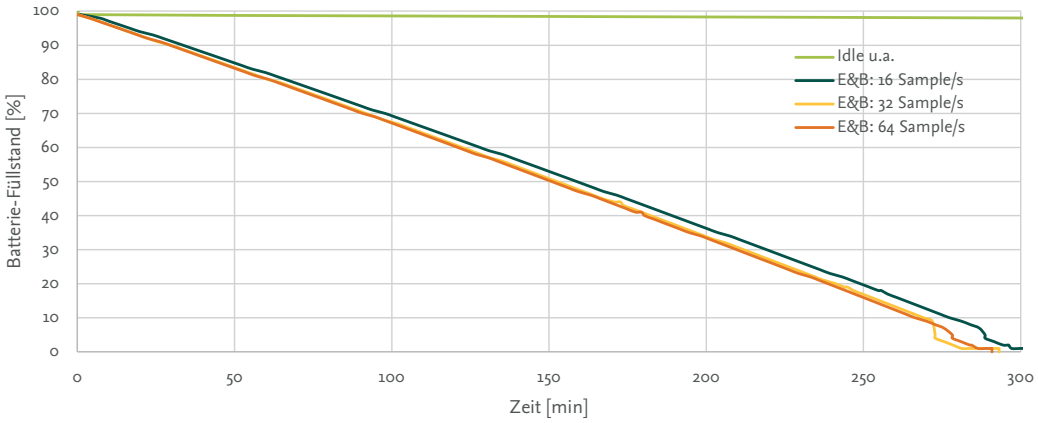


Abbildung 7.4: Spannungsverlauf des Smartphones: Empfangen und Berechnen bei unterschiedlichen Abtastraten.

7.3.3 Messwerte und Sturzerkennung auf Sensorknoten

Wenn der Sensorknoten die Algorithmen zur Sturzerkennung ausführt, ergibt sich für den Energieverbrauch des Smartphones die Kurve „SE auf INGA“ in Abbildung 7.3. Es ist deutlich zu erkennen, dass das Smartphone bei diesem Szenario zur Sturzerkennung die wenigste Energie verbraucht, obwohl die Bluetooth-Verbindung die gesamte Zeit über besteht. Der Vorteil der determinierten Position der Sensorik besteht hierbei weiterhin, jedoch können die Algorithmen zur Sturzerkennung auf weit weniger Rechenleistung zurückgreifen, da in diesem Fall nur INGAs 8-Bit-Mikrocontroller zur Verfügung steht. Dafür lässt sich nun zur Realisierung der Algorithmen die vorhandene „Intelligenz“ des Sensors nutzen, der über mehrere Interrupts bereits die Detektion singulärer Ereignisse ausgeben kann.

Wenn die Sturzerkennung auf dem Sensorknoten berechnet wird, dient das Smartphone lediglich als Relais zum Absetzen der Alarmmeldungen. In diesem Fall würde auch ein (altes) Mobiltelefon ausreichen, welches die Vorteile der längeren Batterielaufzeit und der geringeren Kosten auf sich vereint.

7.3.4 Messwerte und Sturzerkennung auf dem Smartphone

Zum Vergleich wurde die Datenaufnahme und Berechnung der Sturzerkennung auch für die alleinige Verwendung des Smartphones implementiert und evaluiert. Da das Betriebssystem des Smartphones die Verwendung von Interrupts der Sensorik nicht unterstützt, muss der interne Beschleunigungssensor kontinuierlich abgefragt werden. Für den Energieverbrauch des Smartphones ergibt sich die Kurve „SE auf Smartphone“ in Abbildung 7.3: Die Stromaufnahme ist in etwa doppelt so hoch wie bei der Sturzerkennung auf dem Sensorknoten, jedoch lange nicht so hoch, wie bei der kontinuierlichen Übermittlung der Messwerte vom Sensorknoten zum Smartphone.

7.3.5 Energieverbrauch des erweiterten Sensorknotens

Solange der Sensorknoten Teil eines Sturzerkennungsszenarios ist, verbraucht er auch entsprechende Energie. Da auch INGAs Batteriekapazität endlich ist, soll der diesbezügliche Energieanteil an dieser Stelle spezifiziert werden. In Abbildung 7.5 ist der Verlauf der Batteriespannung für die einzelnen Szenarien aufgetragen. Auch hier ist zu erkennen, dass das Samplen und Senden von Daten zum Smartphone („S&S“-Kurven – korrespondierend zu den „E&B“-Kurven des Smartphones) am meisten Energie benötigt. Bei einer Batteriespannung von 3 Volt würde INGA in der aktuellen Konfiguration aufhören zu funktionieren. Beim kontinuierlichen Versenden der Sample ist dieser Wert nach ∼8 Stunden erreicht.

Die Sturzerkennung auf INGA braucht wenig mehr Energie als das „aktive Warten“ des Sensorknotens und funktioniert in der aktuellen Konfiguration mehr als 20 Stunden lang.

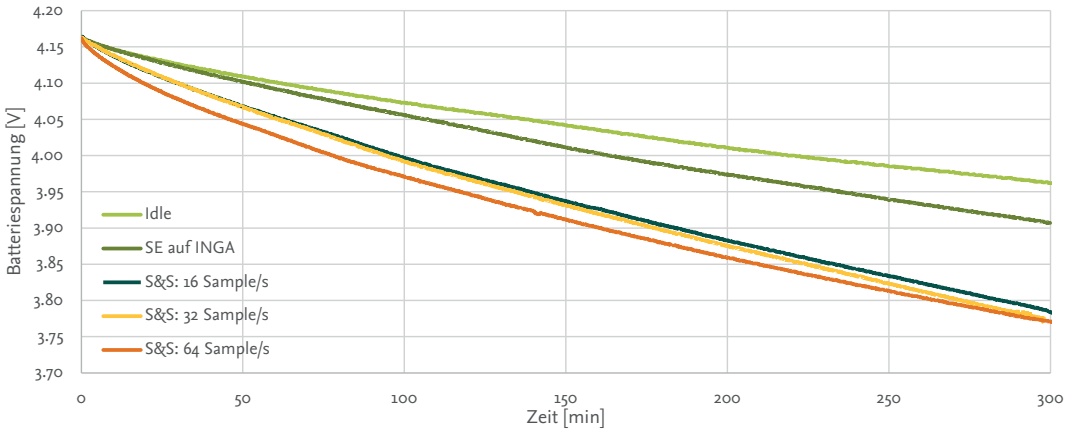


Abbildung 7.5: Spannungsverlauf von INGAs Akku bei unterschiedlichen Implementierungen der Sturzerkennung.

7.4 Fazit: Sturzerkennung und Notfallkanal „*to-go*“

Es war nicht die Absicht dieser Betrachtung, unterschiedliche Algorithmen zur Sturzerkennung auf ihre Qualität und Leistungsfähigkeit hin zu untersuchen. Vielmehr sollte gezeigt werden, dass ein Notrufkanal mit Hilfe eines Mobiltelefons realisierbar ist und dass so die gewünschte Funktionalität eines Notrufsystems, welches sowohl innerhalb als auch außerhalb der Wohnung funktioniert, unterstützt werden kann. Ein solches System kann nicht durch ein einzelnes Gerät realisiert werden, da sich so massive Einschränkungen in der Batterielebensdauer und in der Qualität der Messergebnisse ergeben würden. Vielmehr bedarf es hier der Kombination mehrerer Geräte für den Indoor- und Outdoor-Bereich, wobei der am Körper getragene Messaufnehmer für beide Bereiche derselbe sein sollte, um den Nutzer nicht durch unterschiedliche Systemen zu verunsichern.

Die Wahl der entsprechenden Basisstation erfolgt für den Nutzer idealerweise transparent, das heißt, der Wechsel zwischen Indoor- und Outdoor-System sollte keinerlei Eingriffe bedürfen.

Um die Funktionsfähigkeit zu zeigen, wurde eine Sturzerkennung sowohl auf INGA als auch auf einem Smartphone implementiert und (im Hinblick auf den Energieverbrauch) evaluiert. Dabei hat sich gezeigt, dass die Algorithmen zur Sturzerkennung idealerweise auf dem Sensorknoten zu berechnen sind – zumindest im außer-häuslichen Fall.

Ein Smartphone ist dabei auch nicht unbedingt vorteilhaft zur Datenaufnahme geeignet, da seine Position am Körper nicht festlegbar ist und so unerwartete Messergebnisse auftreten können. Das sporadische Empfangen von Nachrichten über Bluetooth ist für das Smartphone energetisch sogar günstiger, als das dauerhafte Abfragen der eigenen Sensoren. Als Relais zum Absetzen von Notrufnachrichten über Mobilfunknetze funktioniert es jedoch problemlos und sollte in einem mobilen Notrufszenario als solches eingesetzt werden.

8 Infrastruktur-Elemente

> „The major difference between a thing that might go wrong and a thing that cannot possibly go wrong is that when a thing that cannot possibly go wrong goes wrong it usually turns out to be impossible to get at or repair."
>
> Douglas Adams, Mostly Harmless, 1992

Nachdem nun die mobilen Systeme ausführlich behandelt wurden, soll an dieser Stelle ein Blick auf die stationären Systeme geworfen werden, welche aus zwei Stufen bestehen.

Die erste Stufe ist die Basisstation in der häuslichen Umgebung, die sogenannte Multi-Services Home Platform (MSHP). Sie dient als zentrale Instanz für alle handelnden Personen vor Ort und sie verfügt über die benötigten Schnittstellen, um mit allen vorhandenen Sensoren und Aktoren zu interagieren. Mit Hinblick auf das Datenschutzkonzept aus Abschnitt 1.1.6 kommt dieser MSHP eine besondere Rolle zu:

- Hier werden alle personenbezogenen Daten erfasst und gespeichert,
- hier werden Algorithmen zur Aktivitätsbewertung und Sturzerkennung ausgeführt,
- von hier aus werden Alarmmeldungen an Angehörige, Rettungsdienste oder externe Dienstleister versendet.

Für die Basisfunktionalität ist das alleinige Vorhandensein einer MSHP im überwachten Haushalt prinzipiell ausreichend. Mit ihr kann das überwachte Verhalten aufgezeichnet, Stürze können detektiert und Notfallmeldungen abgesetzt werden. Spätestens bei Systemfehlern jedoch kann es sehr sinnvoll sein, zentrale Instanzen im Hintergrund bereitzuhalten, die in der Lage sind, ein Fehlverhalten einzelner MSHPs zu erkennen und eventuell sogar zu beheben. Auch zur Speicherung von Backups und für die Realisierung einer Fernwartung können zentrale Systeme außerhalb der Wohnung sinnvoll sein.

In diesem Kapitel wird deshalb zunächst die MSHP erläutert und die Herausforderungen und Motivationen für die Aspekte Datensicherheit, Datensicherung, Fernüberwachung und Fernwartung dargelegt. Anschließend wird ein integriertes System, welches die genannten Aspekte aufgreift und den Ansprüchen an den Datenschutz genügt, konzipiert, implementiert und evaluiert.

8.1 Multi-Services-Home-Platform – MSHP

Die MSHP stellt die Basisstation im Haus des Nutzers dar, dabei ist eine MSHP immer auch genau einer Person zugeordnet. Damit sichergestellt ist, dass die darauf gespeicherten Daten der „überwachten" Person gehören, sollte sich idealerweise auch die physikalische Plattform im Besitz dieser Person befinden.

Während des Projekts und den Evaluationen wurde die MSHP durch einen Standard-PC (x86-Technologie) realisiert. Es ist aber auch möglich, das System in einer Set-Top-Box zu integrieren [75], was unter Umständen aus Akzeptanz- und Kostengründen sinnvoll sein kann.

In Abbildung 8.1 ist zunächst ein Überblick über die mit verschiedenen Schnittstellen angebundenen Sensoren und Aktoren gegeben, die in den verschiedenen Anwendungsfällen des GAL-Projekts eingesetzt wurden: Ultraschallsensoren zur Lokalisierung von Personen in der Wohnung sind per I^2C angebunden, Mikrofone zur akustischen Steuerung des Systems und zur Aktivitätserkennung per USB. Kameras, die für

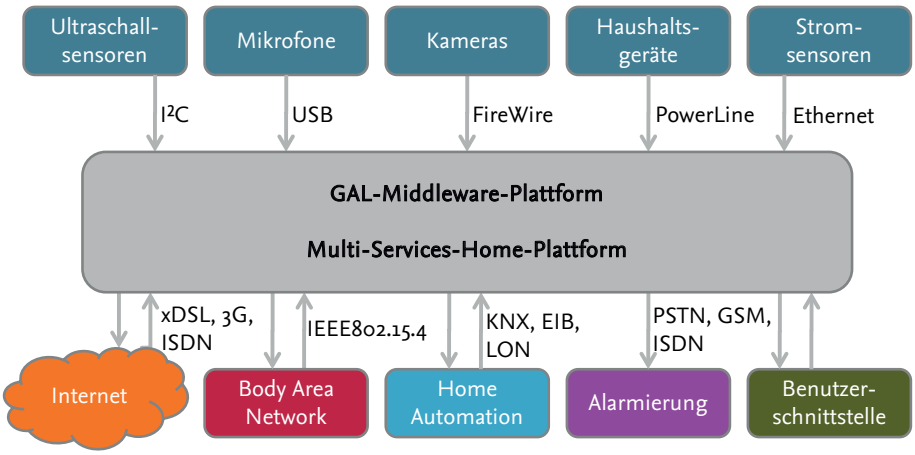


Abbildung 8.1: Überblick über die an die MSHP angeschlossenen Sensoren und Aktoren sowie die verwendeten Schnittstellen.

eine Sturzerkennung und Ganganalyse eingesetzt werden, sind (unter anderem) per FireWire angeschlossen, "intelligente" Haushaltsgeräte über eine PowerLine-Bridge. Zur Aufzeichnung des Stromverbrauchs sind zentrale Stromsensoren per Ethernet angeschlossen, die eine Detektion der benutzen Haushaltsgeräte erlauben und so zusätzliche Informationen zur Bestimmung von Aktivitäten liefern sollen. Benutzerschnittstellen sind zum einen als normaler Tastatur/Maus/Bildschirm-Zugang ausgeführt, zum anderen lassen sich aber auch Smartphones, Tablet-Computer oder eine akustische Steuerung realisieren. Das Versenden von Alarmmeldungen erfolgt entweder über das normale Telefonnetz, über ISDN oder über ein zusätzliches GSM-Modul. Über diverse Schnittstellen (KNX, LON, FS20, etc.) sind Geräte der Heimautomatisierung angebunden, die auf mehrere Arten genutzt werden können: Auf der einen Seite dienen sie als Sensoren zur Aktivitätsbestimmung, auf der anderen Seite auch als Aktoren zur Meldung von Ereignissen. Das WBAN ist über einen externen IEEE 802.15.4-Transceiver angebunden, wobei die in Kapitel 6 vorgestellten Techniken zum Einsatz kommen. Für dieses Kapitel von essentiellem Interesse ist die Anbindung der MSHP an das Internet, welche über diverse Schnittstellen realisiert werden kann (siehe Abschnitt 8.1.2).

8.1.1 Hard- und Software der MSHP

Die Grundlage für alle Implementierungen ist die OSGi-Service-Plattform [157], welche als Middleware für alle implementierten Anwendungen dient [71]. OSGi setzt auf einer JAVA-*Virtual Machine (VM)* auf, was eine gewisse Betriebssystemunabhängigkeit garantiert. In der Praxis ist diese Unabhängigkeit nicht immer gegeben, da die angeschlossenen Sensoren und Aktoren auch Hardwaretreiber für das darunterliegende Betriebssystem benötigen; im GAL-Projekt wurde deshalb Linux (32-Bit) als Zielsystem festgelegt.

Als Hardwareplattform eignet sich im Prinzip jedes System, das in der Lage ist, das entsprechende Betriebssystem, die JAVA-VM und die OSGi-Service-Plattform auszuführen. Einschränkungen ergeben sich aber durch rechen- und speicherintensive Anwendungen, wie die visuelle oder akustische Sturzerkennung (Bild- und Tonanalyse in „Echtzeit").

Abbildung 8.2 gibt dabei zunächst einen exemplarischen Gesamtüberblick über den Aufbau des Systems: Im oberen Bereich sind die unterschiedlichen logischen Akteure, die über verschiedene Benutzerschnittstellen mit dem System interagieren können, dargestellt. Auch die Alarmierungskomponente für Angehörige und Notfalleinsatzkräfte ist dort aufgeführt. Die unterste Ebene des Systems stellt die Hardwareplattform dar; hier werden die verschiedenen Sensoren und Aktoren physikalisch über die unterschiedlichen Schnittstellen angebunden, wobei für jede physikalische Schnittstelle auch ein Treiber für das darüberliegende Betriebssystem

bereitstehen muss.

Neben den Hardwaretreibern und der JAVA-VM sind auch Dateisystem, Datenbank, Datensicherung (siehe Abschnitt 8.3) und VPN-Gateway (siehe Abschnitt 8.4) als Anwendungen im Betriebssystem und nicht innerhalb der OSGi-Plattform realisiert. Dateisystem und Datenbank sind dabei aus Gründen der Performance nicht weiter abstrahiert worden; es wurden stattdessen innerhalb der OSGi-Plattform Schnittstellen geschaffen, die einen effizienten Zugriff gestatten und dabei auch so flexibel sind, dass sie den Einsatz unterschiedlicher Datenbanksysteme erlauben. Die Datensicherungskomponente außerhalb der OSGi-Umgebung zu realisieren, hat den einfachen Grund, dass so auch die komplette OSGi-Plattform gesichert werden kann. Das später erläuterte VPN-Gateway greift direkt auf Betriebssystemkomponenten zu und ließe sich innerhalb der OSGi-Plattform nur sehr schwer realisieren; außerdem ist so auch von außen ein gesicherter Zugriff auf Betriebssystemebene möglich, der es unter Umständen ermöglicht, die OSGi-Plattform neu zu starten, ohne dabei die Verbindung zu kappen.

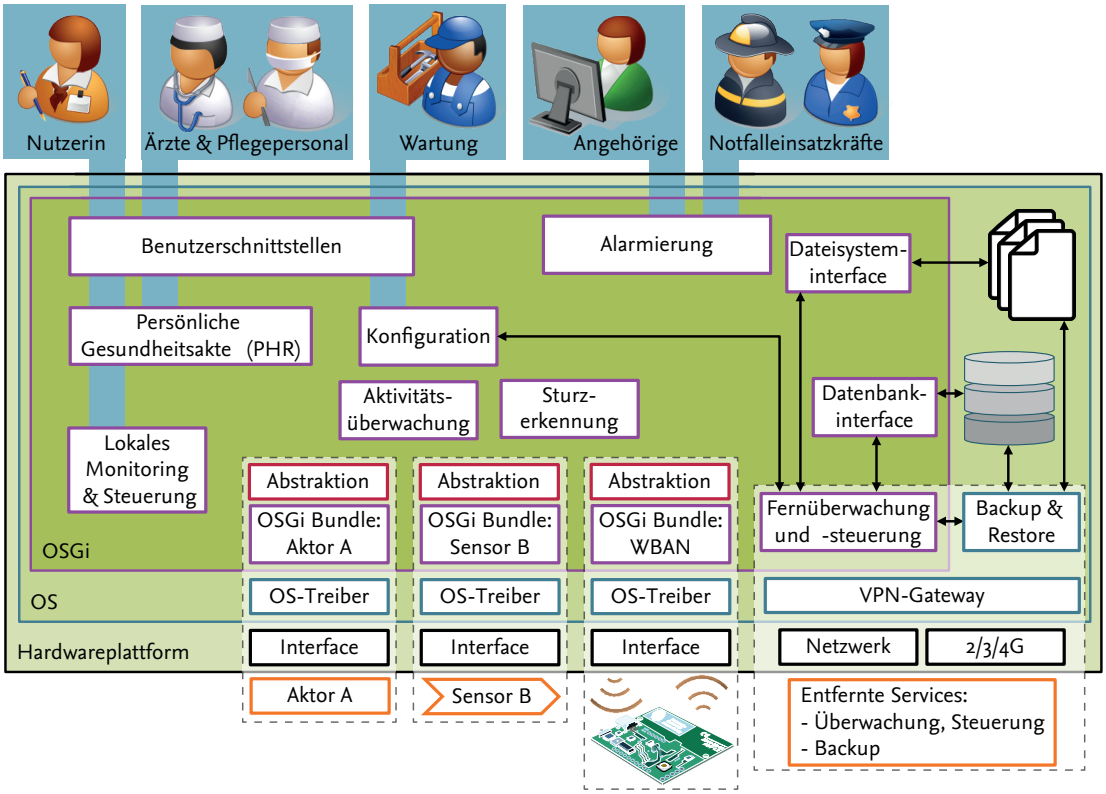


Abbildung 8.2: Systemdiagramm der MSHP mit den implementierten Funktionalitäten, Schnittstellen und beteiligten Akteuren.

8.1.2 Verteilte Systeme

Im Rahmen des GAL-Projekts war es geplant, mehrere reale Wohnungen, in denen Menschen ihr tägliches Leben verbringen, mit der entwickelten Technik auszustatten und diese dort zu evaluieren. Bevor man sich Gedanken macht, wie eine Fernüberwachung oder gar eine Fernkonfiguration solcher Systeme zu bewerkstelligen ist, sollte man die zu erwartende Diversität eventueller Anbindungen an das Internet genauer betrachten.

In Abbildung 8.3 sind deshalb exemplarisch einige der zu erwartenden Gegebenheiten skizziert. Im „Normalfall“ (2) ist eine Internetverbindung über xDSL, ISDN oder DOCSIS realisiert, wobei dem Endkunden in

der Regel eine öffentliche IP-Adresse zur Verfügung gestellt wird und der Heimrouter dann eine Network Address Translation (NAT) auf private IP-Adressbereiche vornimmt; zusätzlich können dabei auch Firewalls (3) zum Einsatz kommen. Je nach Art und Ausprägung eines solchen Anschlusses variieren Durchsatz und Latenz; typischerweise werden bei Heimanwendern asymmetrische Datenraten implementiert, bei denen der Downlink weit höhere Datenraten als der Uplink aufweist. Asymmetrische Übertragungspfade und asymmetrische Datenraten treten auf, wenn der Benutzer über eine Satellitenverbindung mit seinem Internetprovider verbunden ist (4). Der relativ schnelle Downlink verfügt hier über eine relativ hohe Latenz, da er über einen verhältnismäßig weit entfernten Satelliten realisiert wird. Der Uplink hingegen wird meistens über die Telefonleitung realisiert und verfügt über eine geringere Latenz, aber auch über eine weit geringere Bandbreite. Gerade im ländlichen Bereich werden zunehmend Mobilfunknetze zur Bereitstellung eines Internetzugangs genutzt (1); die meisten Anbieter setzen hier eine doppelte NAT ein, um mobile Geräte mit dem Internet zu verbinden.

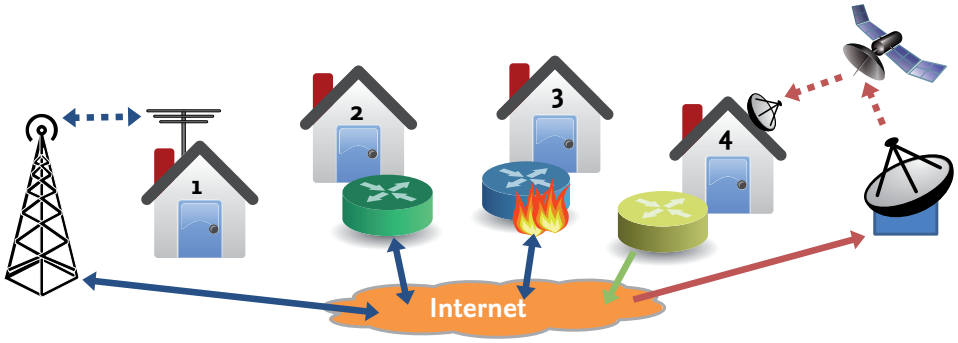


Abbildung 8.3: Mögliche Wege der Anbindung an das Internet: Mobilfunk (1), geroutete öffentliche IP (2), NAT und Firewall (3), asymmetrische Verbindung mit Satelliten-Downlink und Modem-Uplink (4).

8.1.3 Datensicherheit und Maßnahmen zur Erhöhung der Datensicherheit

Die Intention einer Datensicherung im Allgemeinen ist es, einem Datenverlust vorzubeugen und so die Möglichkeit zu haben, gesicherte Daten wiederherzustellen. Ein Datenverlust kann dabei viele Ursachen haben. Im Folgenden sollen diese Ursachen beleuchtet und erläutert werden. Im Sinne des Datenschutzes kann ein „Datenverlust“ sogar gewollt sein, z.B. um nicht unnötig viele Daten zu sammeln, bzw. Daten nach einer bestimmten Zeit oder bestimmten Kriterien explizit zu löschen. Wenn in diesem Abschnitt von „Datenverlust“ die Rede ist, soll jedoch immer ein vom Anwender unbeabsichtigter Datenverlust gemeint sein. Grob lassen sich die Ursachen für einen Datenverlust in Hardwarefehler und Softwarefehler unterteilen. Im Laufe der Auseinandersetzung mit dem Thema fällt jedoch auf, dass eine solche Klassifizierung doch sehr vereinfacht, weswegen unterschiedliche Unterklassen separat beleuchtet werden sollen. Außerdem lässt diese Klassifizierung einen dritten Punkt zunächst völlig außer Acht: Den (berechtigten) Benutzer, der als Mensch Fehler machen kann.

Hardwarefehler

In einem komplexen informationstechnischen System können an unterschiedlichen Stellen Ausfälle auftreten, welche dann wiederum zu einem Datenverlust führen können. Grundsätzlich wird zwischen Komponenten- und Systemausfall unterschieden, wobei ein System aus unterschiedlichen Komponenten besteht. Ein naheliegendes Beispiel für einen Komponentenausfall ist der Ausfall der Festplatte, wobei es zunächst unerheblich ist, wodurch dieser Ausfall verursacht wurde. In einem Standard-PC, wie er zurzeit für die MSHP im Einsatz ist,

kommt der Festplattenausfall zwar einem Systemausfall gleich, da in der Regel nur eine Festplatte verbaut ist, aber dies ist nicht immer der Fall. Ein Systemausfall bedeutet aber nicht zwangsläufig, dass alle vorhandenen Daten „verloren“ sind, auf jeden Fall werden dann aber keine neuen Daten mehr aufgezeichnet.

Jede mechanische und elektrische Komponente unterliegt einem natürlichen Alterungsprozess. Diesen Prozess kann man positiv und negativ beeinflussen -- so beeinflusst eine hohe Temperatur den Alterungsprozess insofern, als dass die erwartete Lebensdauer des Bauteils verkürzt wird. In der Regel fällt die Ausfallrate einzelner Komponenten unter die sogenannte Badewannenkurve [158].

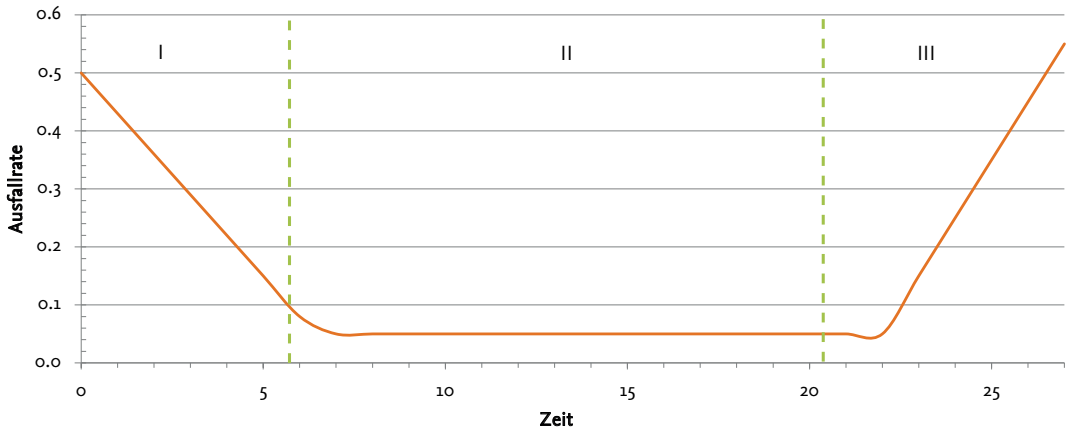


Abbildung 8.4: Typische Badewannenkurve: Viele Ausfälle zu Beginn, wenige während und erneute viele zum Ende der vorgesehenen Benutzungsdauer.

Während in Phase I von Abbildung 8.4 die frühen, meist herstellungsbedingten Ausfälle fallen, treten in Phase II konstante und zufällige Ausfälle in geringer Anzahl auf. Aus diesem Grund sollten alle Komponenten vor dem produktiven Betrieb auch zunächst intensiv getestet werden, damit die Phase I überwunden ist. In Phase III steigt die Ausfallrate wieder stark an: Hier ist das durchschnittliche Ende der Funktionsdauer erreicht. Für viele Hardwarekomponenten (z.B. Festplatten) gibt eine Angabe für die MTBF (Mean Time Between Failures – mittlere Betriebsdauer zwischen Ausfällen) [159] oder die MTTF (Mean Time To Failure – mittlere Betriebsdauer bis zum Ausfall) über die erwartete Lebensdauer einzelnen Komponenten unter Idealbedingungen (keine externen Einflüsse wie Spannungsspitzen oder extreme Temperaturen, kein häufiges Aus- und Einschalten) Auskunft. Natürlich kann ein zufälliger Ausfall in Phase II immer auftreten, weswegen die Angaben zu MTBF und MTTF keine Garantie dafür sind, wie lange eine Komponente tatsächlich funktioniert.

Softwarefehler

Neben defekten Bauteilen oder komplett ausgefallenen/zerstörten Systemen, kann auch das Betriebssystem, bzw. die verwendete Software einen Datenverlust verursachen. Dabei gilt die Regel: Je komplexer das System, desto höher die Fehlerwahrscheinlichkeit. Auch hier sind die Möglichkeiten zahlreich: Angefangen von unwissentlich fehlerhafter Implementierung einzelner Komponenten, über unentdeckt fehlerhaft implementierte Betriebssystemroutinen oder falsch gesetzte Berechtigungen, bis hin zum (un)wissentlich installierten Schadprogramm (Computer-Virus) ist alles denkbar.

Bedienungsfehler

Außerdem kann ein Datenverlust durch simple Bedienungsfehler eintreten: Ein (dazu berechtigter) Nutzer kann die Daten einfach löschen -- ob beabsichtigt oder nicht. In diesem Fall helfen weder gespiegelte Festplatten noch redundant ausgelegte Controller.

Zusammenfassung: Ursachen für Datenverlust und sinnvolle Gegenmaßnahmen

Ein Datenverlust kann viele Ursachen haben – von unterschiedlichsten Hard- und Softwarefehlern über Bedienungsfehler oder absichtliches Löschen/Manipulieren der Daten bis hin zu unwiederbringlich zerstörten oder verlorenen Systemen.

Alles in allem kann man mit einer beliebig großen Menge an Geld das Risiko eines Datenverlustes an bestimmten Stellen des Systems zumindest verringern, wobei einige sicher sinnvoller sind als andere. Einen Bedienungsfehler, wie das versehentliche Löschen von Daten oder aber den „Verlust“ des Gesamtsystems durch Diebstahl oder Feuer können aber auch alle oben genannten Verfahren nicht abdecken.

Eine hundertprozentige Datensicherheit kann in keinem Fall garantiert werden, aber durch eine regelmäßige Datensicherung (Backup, siehe Abschnitt 8.3) zu einem räumlich entfernten System lassen sich die zu erwartenden Verluste erheblich verringern.

8.2 Backend-Systeme

Wie bereits erwähnt, gibt es zahlreiche Gründe, ein eigentlich autark funktionierendes System auch in eine zentrale Struktur einzubinden:

Die installierten Systeme sollen möglichst ohne Benutzerinteraktion funktionieren und sich möglichst ambient verhalten. Im Idealfall bemerkt der Bewohner gar nicht, dass ein MSHP-System mit angeschlossener Sensorik installiert ist. Das führt allerdings auch dazu, dass Systemfehler oder Ausfälle womöglich unbemerkt bleiben. Dieses Verhalten kann sehr wohl gewollt sein, da der Benutzer in der Regel nicht in der Lage sein wird, Systemfehler selbständig zu beheben. Damit der Betreiber solch verteilter Systeme zumindest einen Überblick hat, ob und wie die Systeme funktionieren, braucht es eine Art von Überwachung, die dem Betreiber Informationen zur Verfügung stellt, ob (und in vielen Fällen auch „wie“) ein System funktioniert. Eine Überwachung kann lokal stattfinden, erfordert dann aber auch Personen vor Ort, die in den GAL-Szenarien nicht vorgesehen sind. In anderen Szenarien, wie beispielsweise in einem Seniorenheim, kann die Systemüberwachung unter Umständen auch von ohnehin anwesendem Betreuungs- oder Pflegepersonal durchgeführt werden, wobei dort zusätzliche Aufgaben sicher auch nicht auf Gegenliebe stoßen werden. In der Regel ist daher eine Fernüberwachung sinnvoll und notwendig, die ebenfalls transparent für den Nutzer des Systems stattfindet und die zunächst keine Interaktion erfordert.

Wenn bei einer Überwachung Probleme festgestellt werden sollten, gilt es diese zu beheben. Damit nicht für jedes „kleine“ Problem ein Servicetechniker die entsprechende Wohnung aufsuchen muss, empfiehlt es sich schon aus Gründen der Kostenvermeidung, eine Fernwartungsmöglichkeit der Systeme vorzusehen.

Im Falle eines schwerwiegenden Fehlers, der zu Datenverlust führt, ist ein regelmäßiges Backup dieser Daten eine sinnvolle Möglichkeit, um einem umfangreichen Datenverlust vorzubeugen. Da ein lokales Backup im Falle einer systemweiten Korruption (bspw. durch Brand oder Diebstahl) wenig hilfreich ist, sollten die Backups generell an einem anderen Ort als das zu sichernde System gelagert werden. Alle Techniken zu Fernüberwachung, Fernwartung und Backup betreffen auch Fragestellungen des Datenschutzes.

8.2.1 Systemüberwachung

Bei örtlich verteilten Systemen, wie in Abschnitt 8.1.2 beschrieben, ist es grundsätzlich wünschenswert, von einer zentralen Stelle auf diese Systeme zugreifen zu können. Als zentrales Kommunikationsmedium bietet sich dabei das Internet an, sofern verfügbar. Sollte keine verfügbare Internetverbindung bestehen, ist zu überlegen, ob eventuell durch den Einsatz von GSM/GPRS/3G/4G-Modems eine (temporäre) Internetverbindung zum Zweck der Fernüberwachung hergestellt werden kann.

Fernüberwachung oder *Monitoring* heißt in diesem Fall noch nicht, das System auch zu steuern oder zu konfigurieren — das wäre erst der nächste Schritt. Für einen Systemadministrator wäre sicherlich eine

komplette Kontrolle der zu überwachenden Systeme wünschenswert (root-Zugriff) – das wird jedoch an einigen Stellen mit dem Datenschutzkonzept (siehe Abschnitt 1.1.6) kollidieren. Im Folgenden werden einige Ansätze kurz erläutert.

Heartbeat-Messages

Die Idee hinter einfachen „Herzschlag"-Nachrichten ist relativ simpel: Das überwachte System „meldet" sich in festgelegten Intervallen bei seinem Überwacher. Dieses kann z.B. durch *ping*-(ICMP echo request)-Nachrichten [160] geschehen. Das Empfangen und Senden von *pings* ist in fast jedem netzwerkfähigen Betriebssystem integriert und kann sehr einfach implementiert werden. Wenn diese *pings* von den verteilten Systemen zu einem zentralen Überwachungsorgan gesendet werden, kann dort ein grober Überblick über die verteilten Systeme gegeben werden. Da aber ein *ping* nicht viel mehr Informationen als den Sender und den Empfänger enthält, sind die verwertbaren Informationen auch sehr begrenzt: Die einzigen Aussagen, die ein empfangener *ping* erlaubt sind:

- Das zu überwachende System läuft,
- die Internetverbindung ist vorhanden und
- der ping-sendende Prozess ist noch nicht abgestürzt.

Aber selbst bei einfachen Heartbeat-Nachrichten gibt es schon Sicherheitsrisiken: Da *pings* in Klartext übermittelt werden, ist nicht nur das empfangende System, sondern auch jedes System auf dem Weg in der Lage, diese Nachricht zu empfangen und die (zugegebenermaßen nicht sehr wertvollen) Informationen auszuwerten. Ein größeres Risiko besteht allerdings dadurch, dass das Senden von ICMP-Nachrichten einen Zugang zum Internet (auf IP-Ebene) voraussetzt. Und das wiederum setzt ein gut konfiguriertes, gewartetes und gesichertes System auf dem aktuellsten Stand voraus, welches idealerweise von einer ebenfalls aktuellen Firewall geschützt ist.

Simple Network Management Protocol – SNMP

Weitere Methoden zur Fernüberwachung teilen in der Regel die gerade erwähnten Sicherheitsrisiken. So setzt jede Überwachung einen entsprechend konfigurierten Dienst auf dem zu überwachenden System voraus — und jeder dieser Dienste kann im Laufe der Zeit korrumpiert werden und so ein Einfallstor für Viren, Spionagesoftware und vieles Weiteres darstellen. An dieser Stelle sein nur kurz das Simple Network Management Protocol (SNMP) [161] erwähnt, welches quasi den „Goldstandard" in der Systemdiagnose und –wartung darstellt. Allerdings nur für in-sich-geschlossene Netze und nicht über unsichere Netze wie das Internet, da Sicherheitsfeatures in SNMP komplett fehlen, bzw. erst in der Version 3 des Protokolls vorgesehen sind, welches noch keine größere Verbreitung gefunden hat.

8.2.2 Fernwartung und Systemsteuerung

Das gerade erwähnte SNMP lässt sich auch zur Konfiguration und damit zur Wartung von Hard- oder Softwarekomponenten nutzen. Für einzelne Wartungsaufgaben können entsprechend eingeschränkte Dienste bereitgestellt werden, die genau spezifizierte Aufgaben durchführen, was unter Umständen mit einer entschärften Variante des Datenschutzkonzepts vereinbar wäre. Eine umfassende Fernwartung kann in der Regel nur mit einem Fernzugriff auf root-Ebene geschehen. Hierzu würde sich z.B. eine Remote-Shell wie SSH eignen.

Was für eine Fernüberwachung gilt, gilt für eine Fernwartung um so mehr: Bei der Nutzung von unsicheren Kanälen wie dem Internet, ist erhöhte Vorsicht geboten. Außerdem widersprechen diese Ansätze den Anforderungen des Datenschutzkonzepts (siehe Abschnitt 1.1.6).

Wie dennoch eine Fernüberwachung und in gewissem Maße auch eine Fernwartung implementiert werden können, wird im übernächsten Abschnitt (8.4) gezeigt.

8.3 Datensicherung und Backup

Im Abschnitt 8.1.3 wurde erläutert, wie eine lokale Datenhaltung im Sinne der Datensicherheit verbessert werden kann. Alle diese Maßnahmen haben aber klare Grenzen und es sind immer (einfache) Szenarien denkbar, bei denen noch immer Teile der Daten oder gar alle Daten verloren gehen können. Außerdem widerspricht es auch dem Einsatz in realen Szenarien, wenn eine geringfügige höhere lokale Datensicherheit durch redundante Auslegung kritischer Komponenten mit einem immensen Aufwand an Hardware und damit Geld erkauft wird. Des Weiteren werden durch gespiegelte Festplatten oder Hauptspeicher mit Fehlerkorrektur noch keine Benutzerfehler, wie das unbeabsichtigte Löschen, verhindert. Hierzu sind andere Mechanismen nötig, wie sie im Folgenden erläutert werden.

8.3.1 Backup im lokalen Dateisystem

Die Vorgehensweise bei einem Backup ist es, Datenbestände zu bestimmten Zeiten zu sichern und (idealerweise) unveränderbar zu speichern. Dazu werden Kopien der jeweiligen Daten angelegt und diese an bestimmten Orten abgelegt. Im Falle eines Fehlers zu einem bestimmten Zeitpunkt, z.B. das versehentliche Löschen eines Datensatzes, kann so „in der Zeit zurückgegangen werden“ und so ein Backup von einem Zeitpunkt vor dem Löschen des Datensatzes wieder eingespielt werden. Die Frequenz bzw. Häufigkeit eines Backups entscheidet dabei über die Anzahl und Verteilung dieser „Zeitpunkte in der Vergangenheit“, deren Zustand wiederhergestellt werden kann. Daten, die nach dem letzten Backup angefallen sind, werden allerdings auch von der ausgereiftesten Strategie nicht erfasst — also kann auch ein Backup keine hundertprozentige Datensicherheit garantieren. Um Speicherplatz zu sparen und einen Überblick zu behalten, sollten Backups auch in bestimmten Abständen gelöscht werden. Immer nur die neuesten Backups zu behalten und ältere generell zu löschen, kann bei wenig benutzten Modulen, bzw. spät entdeckten Fehlern, dazu führen, dass in den vorhandenen Backups die gewünschten Daten nicht mehr vorhanden sind. Um dieses zu umgehen, hat sich ein hierarchisches Vorgehen als eine gängige Strategie etabliert: So hält man z.B. sieben tägliche Backups für die vergangenen Tage vor, daneben vier wöchentliche Backups für die vergangenen Wochen und 12 monatliche Backups für die vergangenen Monate und hat so mit 23 Backups ein ganzes Jahr abgedeckt, wobei natürlich die zeitliche Granularität nach hinten gröber wird. Ein solches Backup kann auf mehrere Arten durchgeführt werden. Im Folgenden werden die wichtigsten kurz beschrieben.

Vollbackup

Unter einem Vollbackup versteht man die Sicherung des gesamten Datenbestandes zu einem Zeitpunkt t. Ein Vollbackup ermöglicht die einfache Wiederherstellung des Systemzustands zum Zeitpunkt t, da sich alle benötigten Daten im Vollbackup befinden. Da sich der Datenbestand jedoch meistens kontinuierlich verändert/erweitert und alte Daten meist weiterhin vorhanden sind, wird bei einem häufigen Vollbackup viel Speicherplatz und Übertragungsbandbreite verschwendet. Wenn man z.B. davon ausgeht, dass zum Zeitpunkt t_1 in Abbildung 8.5 eine Menge an x Daten vorhanden ist und zum Zeitpunkt t_2 y Daten hinzukommen und man in beiden Fällen ein Vollbackup durchführen würde, würden zum Zeitpunkt t_2 die Daten x ein zweites Mal gesichert werden und zum Zeitpunkt t_3 ein drittes Mal.

Dieses Vorgehen ist ziemlich sicher, da keine Abhängigkeiten zu anderen (beispielsweise vorherigen) Backups bestehen; sollte das gewünschte Backup korrupt sein, kann auf das vorherige zurückgegriffen werden. Aus Gründen der Effizienz sollten jedoch auch andere und intelligentere Verfahren berücksichtigt werden, die zu weniger Datenaufkommen damit und zu weniger Übertragungsaufkommen führen.

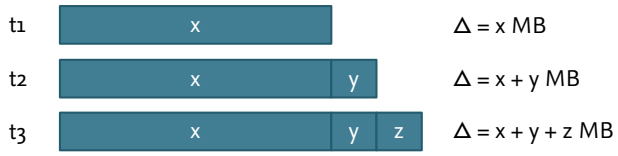


Abbildung 8.5: Vollbackup: Es wird jedes Mal die gesamte Datenmenge gesichert und übertragen.

Differentielles Backup

Ein differentielles Backup geht von einem Vollbackup zum Zeitpunkt *t*1 aus. Darauf aufbauend werden alle Daten, die sich im Bezug auf das Vollbackup geändert haben, gesichert. Wie in Abbildung 8.6 zu sehen, wird also im Bezug auf ein reines Vollbackup Speicherplatz gespart, da der Grundbestand an Daten nicht jedes Mal mitgesichert wird. Allerdings ist die Speichernutzung beim differentiellen Backup auch nicht optimal, da im Beispiel die Daten „y“ doppelt gesichert werden. Will man beispielsweise die Daten zum Zeitpunkt t3 wiederherstellen, so ist zunächst das Vollbackup (t1) und dann das differentielle Backup t3 einzuspielen. Hierbei wird auch ein mögliches Problem deutlich: Wenn das Vollbackup t1 defekt/korrupt sein sollte, sind auch die Daten der darauffolgenden differentiellen Backups wertlos bzw. unbrauchbar, da diese auf das Vollbackup aufbauen.

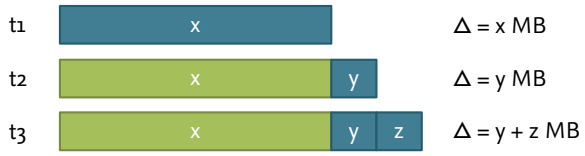


Abbildung 8.6: Differentielles Backup: Es wird jedes die Differenz zum letzten Vollbackup gesichert und übertragen.

Inkrementelles Backup

Bei einem inkrementellen Backup werden nur die Daten gesichert, die nach dem letzten (inkrementellen) Backup geändert wurden, bzw. neu hinzugekommen sind. So werden im Beispiel in Abbildung 8.7 zum Zeitpunkt t3 tatsächlich nur die Daten „z“ gesichert. Bei der Wiederherstellung der Daten zum Zeitpunkt t3 müssen dann aber zunächst alle inkrementellen Sicherungen der vorherigen Zeitpunkte (hier: t1 und t2) eingespielt werden. Neben diesem Mehraufwand besteht gegenüber der differentiellen Sicherung ein größeres Sicherheitsrisiko, da nun alle inkrementellen Sicherungen bis zum gewünschten Zeitpunkt intakt sein müssen.

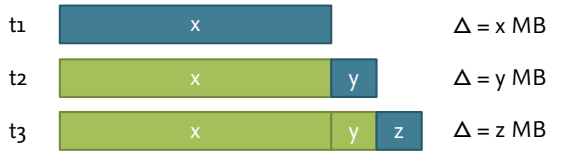


Abbildung 8.7: Inkrementelles Backup: Es wird nur die Differenz zum letzten (inkrementellen) Backup gesichert und übertragen.

8.3.2 Lagerung der Sicherungen

Die gerade vorgestellten Backup-Strategien für lokale Dateisystem-, bzw. lokale Datenbank-Backups haben gemein, dass sie zunächst alle auf dem Ursprungssystem lagern. So können zwar Fehler wie unbeabsichtigtes Löschen behandelt werden, aber die „gesicherten“ Daten sind noch immer genauso anfällig für alle in Abschnitt 8.1.3 vorgestellten Möglichkeiten an Hardwarefehlern, die sich so gleichzeitig auf originale Daten und vermeintlich gesicherte Daten auswirken können. Um diese Fehlermöglichkeit sinnvoll zu minimieren und gleichzeitig auch Totalverluste durch Feuer, Wasser oder Diebstahl zu behandeln, bietet sich eigentlich nur die Lagerung der Sicherungsdaten auf einem örtlich entfernten Datenträger oder System an. Der geplante Einsatz des Systems als autonom fungierendes Gateway und die zu erwartende geringe Technikkompetenz der Nutzer schließt eine Sicherung, bei der eine Anwenderinteraktion erforderlich ist, aus. So kann es von den Nutzern der MSHP nicht erwartet werden, täglich USB-Sticks zu tauschen oder Festplatten zu wechseln. Eigentlich kommt auch eine Sicherung über das Internet auf örtlich entfernte Systeme nicht infrage, da hierbei das in GAL geforderte Prinzip der Datenhoheit des Nutzers übergangen wird. Wie es dennoch möglich sein kann, eine Sicherung über das Internet zu realisieren, wird konkret im nächsten Abschnitt behandelt.

8.3.3 Datensicherungskonzept

Nachdem in den vorherigen Abschnitten die Notwendigkeit und die Grundlagen für eine Datensicherung erläutert wurden, wird nun auf die konkreten Anforderungen eingegangen und ein schlüssiges und sinnvolles Datensicherungskonzept aufgezeigt. Dabei kommen zuvor erläuterte Techniken zum Einsatz. Da die MSHP, um sinnvoll eingesetzt werden zu können, auch über eine Internetverbindung angebunden sein muss, liegt es nahe, diese auch für die Datensicherung zu verwenden. Eine in einem Privathaushalt vorhandene Internetverbindung wird jedoch nicht so dimensioniert sein, dass darüber täglich große Mengen an Daten für ein Vollbackup (s.o.) der kompletten MSHP verschickt werden können.

Der Grundsatz, dass alle Daten stets unter der Kontrolle des Nutzers bleiben, wird durch eine Datensicherung auf nicht-lokalem (also entferntem) Speicherplatz ausgehebelt. Das Datenschutzkonzept (siehe Abschnitt 1.1.6) sieht vor, dass die Daten die MSHP niemals ohne Zustimmung des Nutzers verlassen dürfen. Für eine automatisierte Backup-Strategie, welche eine Speicherung auf räumlich entferntem Speicherplatz vorsieht, ist das jedoch sehr hinderlich. Auch ist es grundsätzlich nicht wünschenswert, dass die Daten dann für dritte einsehbar auf irgendwelchen Servern liegen -– auch wenn die „dritten“ nur Administratoren der Server sind, ist immer davon auszugehen, dass, wenn die Daten erst einmal irgendwo zugänglich lagern, diese dann auch Begehrlichkeiten wecken. Es ist also naheliegend und praktikabel, die Daten zu verschlüsseln. Die Verschlüsselung der Daten bringt jedoch auch Nachteile mit sich: So ist es z.B. nicht möglich eine differentielle oder inkrementelle Datensicherung mit verschlüsselten Daten durchzuführen, da die Differenz zum Vorherigen nur in unverschlüsselter Form ersichtlich ist. Um ein inkrementelles Backup zu ermöglichen, ist also das Vorhandensein eines unverschlüsselten Backups auf der MSHP von Nöten. Eine verschlüsselte Kopie dieses Backups -– so wie auch aller weiteren inkrementellen Backups — kann dann an einen Storage-Provider versendet werden.

Wie in Abbildung 8.8 zu erkennen ist, ist es notwendig, dass die unverschlüsselten Backups für ein inkrementelles Backup auf der MSHP verbleiben müssen, da sie die Quelle darstellen, zu der eine Differenz gebildet wird. Die verschlüsselten (inkrementellen) Backups können nach dem Versenden gelöscht werden, da sie für einen Vergleich nichts nützen und nur Speicherplatz brauchen. Um die Datenschutzansprüche zu erfüllen, ist ein sicherer Schlüssel pro MSHP-Installation zu wählen und dieser Schlüssel darf nur auf der jeweiligen MSHP gespeichert sein; nur so kann sichergestellt werden, dass die verschlüsselten Daten nur von dem berechtigten Nutzer wiederhergestellt werden können. Um auch im Falle eines Totalverlusts der MSHP die entfernt gespeicherten Backups wieder einspielen zu können, kann es sinnvoll sein, den Schlüssel auch

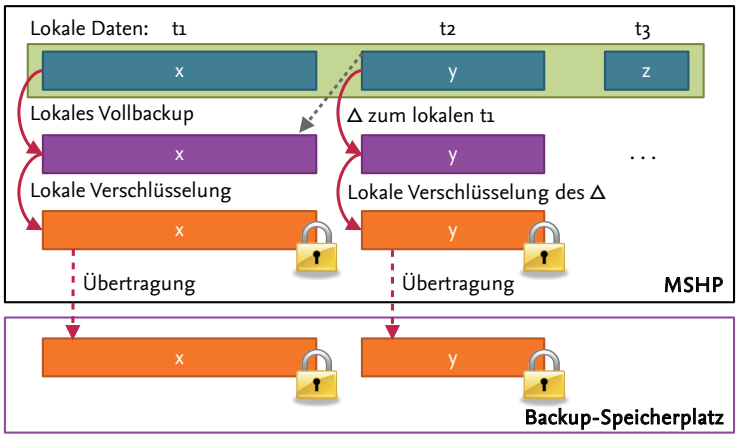


Abbildung 8.8: Die einzelnen Zustände des Backups: Die vollen, differentiellen oder inkrementellen Backups werden lokal erzeugt, verschlüsselt und anschließend übertragen.

noch an einem (gut gesicherten) anderen Ort aufzubewahren, beispielsweise auf einem USB-Stick oder einer Smartcard.

8.3.4 Zusammenfassung: Datensicherung und Backup

Dass eine Datensicherung durch regelmäßige Backups essentiell ist und einem (wie auch immer verursachten) Datenverlust vorbeugen kann, steht außer Frage. Die Speicherung dieser Backups an einem entfernten Ort trägt dazu bei, dass auch bei einem vollständigen System- oder gar Wohnungsverlust eine Wiederherstellung der Daten möglich ist. Das Versenden der Daten zu anderen Systemen birgt aber immer auch Sicherheitsrisiken, weswegen der erste Schritt eine vollständige Verschlüsselung dieser Daten ist – mit einem Schlüssel, der nur dem Besitzer der Daten bekannt ist.

Prinzipiell stellt aber auch das „Abladen“ verschlüsselter Daten bei anonymen Storage-Providern ein Sicherheitsrisiko dar, denn es hat sich im Laufe der Geschichte kaum eine Verschlüsselungsmethode als unknackbar erwiesen (siehe Abschnitt 6.2). Selbst wenn die aktuell verwendete Verschlüsselung als sicher gilt (im vorgestellten Fall kam AES im CBC-Modus mit einer Schlüssellänge von 128 Bit zum Einsatz), kann das in Zukunft anders aussehen. Verschlüsselte Backup-Daten, die bei einem Speicherplatz-Provider liegen, sollten sich deshalb auch garantiert löschen lassen, um nicht zu einem späteren Zeitpunkt von Dritten (mit gesteigerter Rechenleistung oder verbesserten Angriffsmethoden) entschlüsselt werden zu können. Aus diesen Gründen kann es sinnvoll sein, auch den Backup-Speicherplatz im Gesamtsystem vorzusehen und sich nicht auf die Angebote Dritter zu verlassen.

Außerdem müssen für das Speichern der verschlüsselten Daten entsprechende Schnittstellen geschaffen werden, die unter Umständen auch für einen böswilligen Angriff genutzt werden können. Im folgenden Abschnitt wird deshalb eine Möglichkeit beschrieben, mit der sowohl das Monitoring als auch das Backup über einen gesicherten Kanal ablaufen können.

8.4 Datenübertragung zu Backends

Es ist sicherlich möglich, die Firewall eines zu überwachenden MSHP-Systems für nur ausgehende ICMP-Nachrichten zu konfigurieren, doch dies ist nicht sehr flexibel. Umfangreichere Monitoring-Szenarien oder auch eine Fernwartung erfordern eine Lösung, die nicht bei jeder neuen zu überwachenden Instanz neu konfiguriert werden muss. Aus Sicht des Datenschutzes ist es sinnvoll, dass sich niemand aktiv mit einer der verteilten MSHP-Installationen verbinden kann und so Schaden jedweder Art anrichten kann. Es liegt also

nahe, dass die verteilten MSHP-Systeme selber nicht in der Lage sein sollten, überhaupt irgendwelche Verbindungen von der Außenwelt anzunehmen. Um dennoch mit den entfernt installierten MSHPs kommunizieren zu können, muss der Verbindungsaufbau dafür von der jeweiligen MSHP erfolgen:

- Jede Verbindung zwischen Überwachungsinstanz und einer MSHP sollte also von der MSHP selbst initiiert sein.
- Anfragen zu bestimmten Diensten von außen sollten schlichtweg ignoriert werden und ins Leere laufen.

Nur, wie verträgt sich das mit der Fernüberwachung und der Fernwartung? Es verträgt sich sogar sehr gut, denn auch die unterschiedlichen Anbindungen an das Internet (siehe Abbildung 8.3) würden einiges an Problemen aufwerfen: So müssten in den verschiedenen Einzelfällen jeweils Firewalls durchbrochen, Router konfiguriert und Weiterleitungen eingerichtet werden, wobei dies dazu führte, dass die Systeme dadurch angreifbarer (weil exponierter) sind. Ein System hinter einer Firewall oder NAT kann in der Regel nur auf dem Rückweg erreicht werden — nämlich wenn das System zunächst selbständig eine Verbindung "nach draußen" aufgebaut hat. Also sollte eine Verbindung mit einem zu überwachenden System nicht nur aus Sicht des Datenschutzes sondern auch aus Sicht des Netzwerkes nur durch das zu überwachende System initiiert sein. Diese beiden Gegebenheiten drängen geradezu zum Einsatz eines VPNs, welches durch die jeweils zu überwachende/konfigurierende MSHP initiiert wird.

8.4.1 Virtuelle Private Netzwerke (VPNs)

Ein virtuelles privates Netzwerk (Virtual Private Network (VPN)) erstellt einen (sicheren) Tunnel durch ein unzuverlässiges und unsicheres Netzwerk wie das Internet. Allen VPN-Lösungen ist gemein, dass die Daten am Eingang zum Tunnel verpackt werden und am Ausgang wieder entpackt werden. In [162] werden die gängigsten Ansätze vorgestellt und verglichen. Das *Layer 2 Transport Protocol (L2TP)* [163], die Internet Protocol Security (IPsec) [164] und das *Point-to-point Tunneling Protocol (PPTP)* [165] sind etablierte und ziemlich alte Internetstandards. Microsoft, der Entwickler von PPTP, rät inzwischen aufgrund von Sicherheitsmängeln von der Verwendung von PPTP ab. L2TP an sich ist zunächst nur für den Tunnelaufbau und nicht für eine Verschlüsselung zuständig, weswegen es häufig in Verbindung mit IPsec zum Einsatz kommt. IPsec wiederum ist in der Netzwerkschicht (Layer 3) beheimatet und kann deshalb nicht immer ohne weiteres in vorhandene Netzwerke integriert werden; besonders bei wechselnden oder mobilen Netzen sind L2TP und IPsec sehr aufwändig in Installation und Wartung und so nicht gerade fehlerresistent. Peer-to-Peer-VPNs wie beispielsweise tinc [1] werden einem zentralen Monitoring-Ansatz nicht gerecht.

Nach einem intensiven Review der vorhandenen VPN-Lösungen wurde OpenVPN [2] als der am besten geeignete Kandidat identifiziert. OpenVPN ist ein sogenanntes Application-Layer-VPN, es setzt also auf der Anwendungsebene auf. Es benutzt Transport Layer Security (TLS) [166] und nutzt die freien OpenSSL-Bibliotheken [3].

Abbildung 8.9 verdeutlicht die Funktionalität von OpenVPN: Die OpenVPN-Anwendung stellt ein virtuelles Netzwerkinterface zur Verfügung, das sich im System genauso wie ein „normales" Ethernet-Interface verhält. Jeglicher Netzwerkverkehr, der über dieses virtuelle Netzwerkinterface geleitet wird, wird „eingepackt" und über einen sicheren Tunnel weitergeleitet. Dieser Tunnel wird zwischen dem OpenVPN-Client auf der MSHP und einem Server, dem VPN-Concentrator, aufgebaut. Dieser Concentrator enthält ebenfalls ein virtuelles Netzwerkinterface (pro MSHP), welches den anderen Tunnelendpunkt darstellt. Da eine Kommunikation zwischen unterschiedlichen MSHPs aus Gründen des Datenschutzes nicht möglich sein soll, ist der Con-

[1] http://www.tinc-vpn.org
[2] http://www.openvpn.net/
[3] http://www.openssl.org/

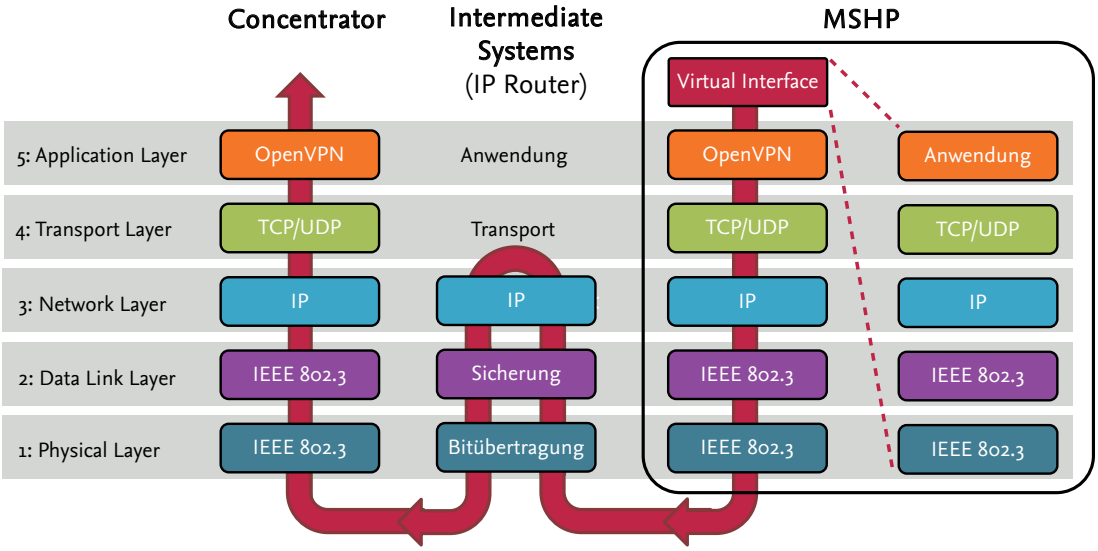


Abbildung 8.9: Die Funktionsweise von OpenVPN: Alle Daten, die an das virtuelle Interface gesendet werden, werden verschlüsselt bis zur zentralen Instanz (Concentrator) getunnelt.

centrator entsprechend als "nicht-routend“ konfiguriert. Für jede zu überwachende MSHP sind so nur zwei Regeln zu beachten:

- Von „draußen“ sollte kein Dienst erreichbar sein. Die einzige Applikation, die mit dem physischen Netzwerkinterface kommunizieren darf, ist der OpenVPN-Prozess – und dieser auch nur für ausgehende Verbindungen.
- Alle anderen Überwachungs-, Konfigurations- und Managementprozesse werden nur an das virtuelle Interface gebunden und können auf diese Weise nur mit dem VPN-Concentrator und nicht mit anderen Systemen im Internet kommunizieren.

Nachdem diese Konfiguration erfolgreich eingerichtet wurde, funktioniert sie über (fast) jede Art von Internetverbindung, da OpenVPN sich auf Anwendungsebene ähnlich einem Webbrowser verhält und so in der Regel keine Firewall- oder sonstige Regeln für die weiterleitenden Systeme eingerichtet werden müssen. Das erlaubt auch eine Vorkonfiguration der Systeme: Mit vorinstallierten Zertifikaten (ein Paar pro MSHP-Concentrator-Verbindung) können die Dienste ohne weitere Nutzerinteraktion starten. Sobald die MSHPs eingeschaltet und ans Internet angeschlossen werden, bauen sie selbstständig eine Verbindung zum VPN-Concentrator auf und die Fernwartung kann dort gestartet werden.

8.5 Zusammenfassung: Fernüberwachung, Fernwartung und Backup über VPN

Sobald über eine VPN-Verbindung ein sicherer Tunnel etabliert wurde, kann darauf jedwede Art von Protokoll und Funktionalität sicher betrieben werden. Das können je nach Umfang des Monitorings zunächst nur simple und unidirektionale Heartbeat-Nachrichten sein. Aber selbst das vermeintlich unsichere SNMP lässt sich über den gesicherten Tunnel zuverlässig über große Distanzen betreiben.

Über SNMP sind dann auch einfache bis komplexe Konfigurationsaufgaben wahrnehmbar. Eine umfangreiche Fernkonfiguration und –wartung lässt sich durch eine SSH-Verbindung realisieren: Indem ein entsprechender SSH-Daemon auf den MSHPs an das virtuelle Interface gebunden wird, werden auch diese

Verbindungen zusätzlich durch den Tunnel gesichert und es muss auf den einzelnen MSHPs kein SSH-Dienst nach außen angeboten werden.

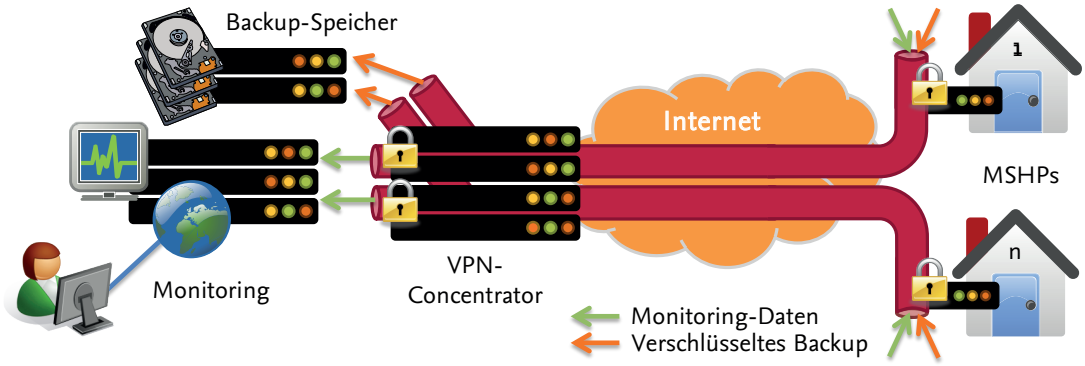


Abbildung 8.10: Funktionalität des Gesamtsystems: Fernüberwachung, Fernwartung und Backups über einen gesicherten VPN-Tunnel pro MSHP.

In Abbildung 8.10 ist die Funktionalität des Fernüberwachungs-, -wartungs- und -backup-Systems zusammenfassend dargestellt. Die verteilten MSHP-Systeme bauen selbstständig jeweils einen VPN-Tunnel zum VPN-Concentrator auf. Dieser Tunnel kann als sicher und zuverlässig angesehen werden, solange keine andere Instanz in den Besitz der Zertifikate gelangt ist. Nur der VPN-Client der MSHP ist in der Lage, das externe Netzwerkinterface anzusprechen; jegliche Kommunikation nach außen wird über den Tunnel geführt und endet am VPN-Concentrator. Am Concentrator sind dann die entpackten und entschlüsselten Monitoring-Daten der einzelnen Clients vorhanden und können beispielsweise an eine Überwachungsinstanz weitergeleitet werden. Diese sollte dann natürlich genauso wie auch der Concentrator in einem gesicherten Raum stehen und nur lokal angebunden sein. Das Backup, welches die verschlüsselten inkrementellen Datensicherungen zu einem Speicherplatz transferiert, kann ebenfalls auf diesen gesicherten Kanal zurückgreifen.

Um die Systeme kontrolliert auszubringen, wurden Mechanismen evaluiert und implementiert, die das automatische Aufsetzen von Betriebssystem und Softwarekomponenten unterstützen. In diesem Zug werden auch Zertifikate (X509 – jeweils ein Paar pro MSHP) erzeugt und entsprechend auf MSHP und Concentrator gespeichert (siehe Abbildung 8.11). Mit dem Zertifikat und der IP-Adresse kann die MSHP nun jederzeit selbstständig und ohne Nutzerinteraktion eine VPN-Verbindung zum Concentrator aufbauen. Durch die Vorinstallation der Zertifikate wird so auch eine Plug-and-Play-Funktionalität erreicht: Sobald das fertig aufgesetzte System an seinen Bestimmungsort gebracht wurde, kann es über jede zur Verfügung stehende Internetverbindung eine Verbindung aufbauen und ist ab diesem Zeitpunkt (je nach Konfiguration) aus der Ferne überwach- oder gar konfigurierbar.

Die initiale Installation der Zertifikate kann dabei aus Gründen des Komforts erfolgen, ist aber nicht zwingend notwendig. Ein späteres manuelles Aufspielen oder Austauschen ist problemlos möglich.

8.6 Implementierung und Evaluation des Systems

Das System wurde wie beschrieben implementiert und konnte seine grundsätzliche Funktionalität unter Beweis stellen. Die zentrale Überwachungsinstanz und der VPN-Concentrator wurden dazu als eigene Server mit öffentlichen IP-Adressen im Universitätsnetz betrieben. Die feste und öffentliche IP-Adresse macht in diesem Fall auch die Nutzung von Nameservern, welche von Dritten betrieben werden, überflüssig, was sonst eine weitere Quelle für Kompromittierungen ausschließt. OpenVPN wurde im *routing*-Modus konfiguriert, wobei explizit keine Routen angegeben wurden, damit die einzelnen Instanzen der MSHPs nicht in der Lage

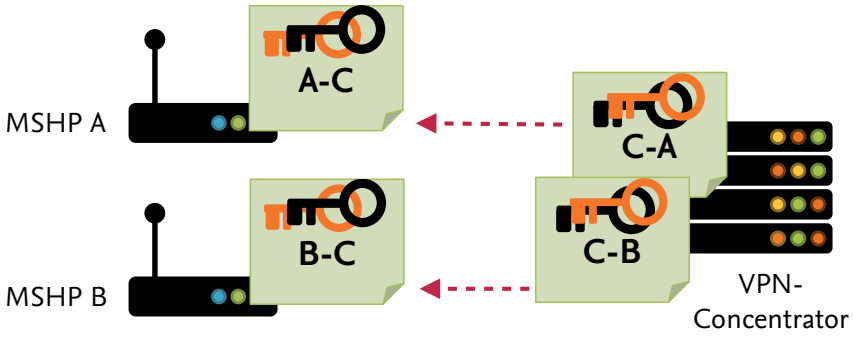


Abbildung 8.11: Verteilung der Zertifikate durch den Concentrator, bevor die einzelnen Systeme ausgebracht werden.

sind, sich untereinander zu erreichen; so kann ausgeschlossen werden, dass kompromittierte MSHPs andere Systeme beeinflussen. Die Schlüsselpaare für jede MSHP-Installation wurden zentral vom Concentrator erzeugt; vor dem „Ausbringen“ der jeweiligen MSHP wurde das dazugehörige Zertifikat auf ihr installiert. Für den Fall, dass ein böswilliges Fehlverhalten bei einer MSHP entdeckt wird, kann im Concentrator das entsprechende Zertifikat gelöscht werden, was weitere Verbindungsversuche dieser MSHP vereitelt.

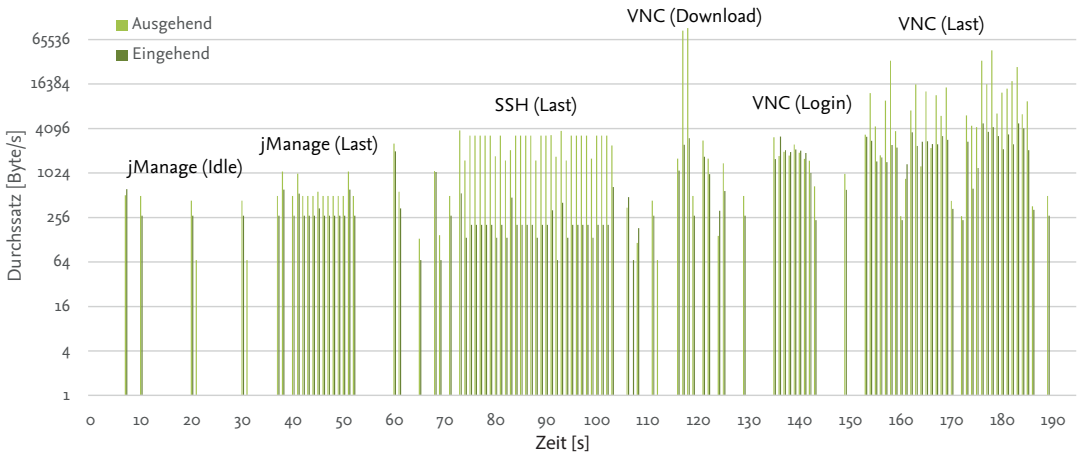


Abbildung 8.12: Durchsatz auf der Netzwerkverbindung zwischen MSHP und Concentrator bei unterschiedlichen Tätigkeiten von Monitoring und Fernwartung.

Das System hat mit jeder getesteten Internetverbindung funktioniert, dabei konnten aber durchaus Einschränkungen in Bezug auf Stabilität und Geschwindigkeit der Internetverbindung festgestellt werden. In Abbildung 8.12 ist die gemessene benötigte Bandbreite für verschiedene Fernsteuerungs- und Monitoringansätze aufgetragen. Das simple Versenden von ICMP-Echo-Request-Nachrichten fällt dabei unter die „Wahrnehmungsschwelle“ dieser Grafik – dies sollte also mit jeder auch noch so schmalbandig ausgelegten Internetverbindung problemlos funktionieren. Auch eine SSH-Session brauchte (für den ungewöhnlichen Fall, dass konstant Daten übermittelt wurden) im Maximum nur eine Bandbreite von 2.8 kByte/s, was selbst bei Annahme einer Modemverbindung funktionsfähig sein sollte. Bei der Verwendung von grafischen Werkzeugen zeigen sich erste – wenn auch nicht sehr schwerwiegende Einschränkungen: Bei der Benutzung von VNC [167] und dem Weiterleiten eines 800x600 Pixel großen Videobildes, werden im Maximum 51.2 kByte/s gemessen, der Durchschnitt liegt hier bei 13.1 kByte/s. Diese Datenraten lassen sich mit einer herkömmlichen Modem-, ISDN- oder GSM-Verbindung nicht erreichen, weswegen für Standorte, die über diese Netze angebunden

sind, von einer grafischen Fernadministration abzusehen ist. Open-Source-Tools wie jManage[4] bieten aber zum Beispiel grafische Management-Applikationen, die auf dem überwachenden Client laufen und so auch verhältnismäßig wenig Bandbreite beanspruchen (im Mittel unter 1 kByte/s).

Da der Durchsatz innerhalb des VPNs gemessen wurde, ist in Abbildung 8.12 der Overhead der VPN-Verbindung nicht mit erfasst. Je UDP-Paket kommen für das Tunneln und Verschlüsseln bei OpenVPN 69 Byte an Overhead hinzu, in realen Anwendungen liegt der Overhead im Mittel bei ∼10 %.

Ein *ping* für die Heartbeat-Nachrichten benötigt brutto 81 Byte. Würde man ihn sekündlich absenden, würde das (inklusive der VPN-Verbindung) ein tägliches Datenaufkommen von knapp 7 MByte verursachen. Während das bei xDSL- oder DOCSIS-Anschlüssen kein Problem darstellt, wäre ein in manchen Mobilfunktarifen inklusives Übertragungsvolumen von 200 MByte pro Monat damit bereits knapp überschritten. Da in der Regel eine Reaktion innerhalb von Sekunden nicht nötig ist, kann eine minütliche Heartbeat-Nachricht das Datenaufkommen auf ∼3.5 MByte pro Monat reduzieren.

Über die vorgestellte, implementierte und evaluierte Anbindung der MSHPs über ein VPN und einen VPN-Concentrator können sowohl Fernüberwachung und Fernwartung als auch Backups durchgeführt werden. Dabei reichen für einfache (nicht-grafische) Überwachungs- und Fernsteuerungsfunktionen die Bandbreite und das Übertragungsvolumen eines jeglichen vorhandenen Internetanschlusses aus. Bei der Übermittlung von grafischen Inhalten oder bei Durchführung von umfangreichen Backups sollte allerdings ausreichend Bandbreite und Volumen zu Verfügung stehen. Gerade in ländlichen Gebieten kann dies zur Zeit noch ein Hinderungsgrund sein, diese Techniken dort einzusetzen. Mit zunehmender Verbreitung von Long Term Evolution (LTE) sollten sich aber auch auf dem Land diese Gegebenheiten ändern: Neben den bereits (selbst über GSM-Netze) funktionierenden Monitoring-Ansätzen werden sich dort in Zukunft auch umfangreiche Fern-Backups nutzen lassen.

[4] `http://www.jmanage.org`

9 Fazit

> „We live in a society exquisitely dependent on science and technology, in which hardly anyone knows anything about science and technology."
>
> Carl Sagan, „Why We Need To Understand Science", 1990

Vor dem Hintergrund einer immer älter werdenden Gesellschaft wurden unter anderem im GAL-Projekt zahlreiche Lösungsansätze erarbeitet, die es alleinstehenden älteren Personen erlauben sollen, längere Zeit sicher in der eigenen Wohnung leben zu können. Dabei setzten viele der behandelten Szenarien die Verfügbarkeit von Daten, die am menschlichen Körper aufgenommen wurden, voraus; seien es Vitalparameter oder Aktivitäts- und Bewegungsdaten. Je nach Anwendungsbereich müssen diese Daten entweder sofort ausgewertet oder für eine spätere Auswertung lokal gespeichert werden. Wenn mehrere unterschiedliche Anwendungsfälle gleichzeitig abgedeckt werden sollen, erscheinen diese Anforderungen zunächst konträr, jedoch konnte in dieser Arbeit gezeigt werden, dass sie sich durchaus vereinen lassen. Die Beiträge dieser Arbeit verteilen sich dabei auf unterschiedliche Aspekte eines Systems, welches mit der Datenerfassung am menschlichen Körper beginnt, die Datenübertragung in der Wohnung bzw. unterwegs beinhaltet und mit der Datenspeicherung auf einem Backend endet.

9.1 Beiträge

Die in dieser Arbeit geleisteten Beiträge sind drei physikalischen Orten zuzuordnen:

1. Den am Körper getragenen drahtlosen Sensorknoten,
2. den häuslichen Gateways und
3. den zentralen Backend-Systemen.

Auf letzteren wurden Mechanismen zu Fernüberwachung und -wartung von divers verteilten häuslichen Gateways implementiert und evaluiert. Dabei stellt ein VPN zwischen diesen zentralen Instanzen und den einzelnen Basisstationen einen sicheren Kommunikationskanal zur Verfügung, der durch die häuslichen Gateways initiiert wird und so eine Plug-and-Play-Lösung für das Ausbringen der verteilten Systeme darstellt.

Für die häuslichen Gateways wurden Funktionalitäten zum Aufbau ebendieses Übertragungskanals implementiert und evaluiert. Außerdem wurden Konzepte zur Datensicherung umgesetzt, die auch über die VPN-Verbindung genutzt werden können und so eine sichere Möglichkeit für Transfer und Lagerung der persönlichen Daten bieten.

Mit dem drahtlosen Sensorknoten INGA wurde ein System geschaffen, das in der Lage ist, die für die skizzierten Anwendungsfälle relevanten Daten aufzunehmen, zu verarbeiten und zu transferieren. Die Datenübertragung zum Gateway wurde dabei über ein unterbrechungstolerantes Kommunikationsprotokoll gelöst, welches es auch ermöglicht, mehrere Anwendungsfälle mit unterschiedlichen Anforderungen gleichzeitig zu bedienen. Eine effiziente Datenreduktion wird durch die Nutzung einer angepassten Deltakodierung erreicht. Mit einem Verschlüsselungssystem, welches auf One-Time-Pads beruht, kann eine nachhaltig sichere Übertragung der Daten zwischen Sensorknoten und Gateway erreicht werden. Um auch außerhalb der Kommunikationsreichweite der Basisstation das Versenden von Notfallmeldungen zu unterstützen, wurde

ein mobiler Notfallkanal implementiert und evaluiert, der ein Smartphone als Gateway zum Mobilfunknetz nutzt.

Die einzelnen Systeme wurden auf unterschiedlichen Ebenen und teilweise im realen Feldeinsatz evaluiert und konnten so ihre Funktionalität und Leistungsfähigkeit zeigen.

9.2 Ausblick

Prinzipiell sind die vorgestellten Ansätze nicht auf die eingangs beschriebenen Szenarien beschränkt. So lassen sich einzelne der erbrachten Beiträge auch in einem anderen Kontext sinnvoll nutzen. Besonders gilt das für die in Kapitel 6 präsentierten Techniken zur Datenübermittlung zwischen mobilem Knoten und Datensenke:

Dass sich DTNs prinzipiell für Anwendungsfälle in WSNs eignen, in denen Daten gesammelt werden, kann an vielen Beispielen festgemacht werden. Häufig kommen bei solch verzögerungs- oder unterbrechungstoleranten Szenarien allerdings für den jeweiligen Anwendungsfall entwickelte Protokolle auf Anwendungsebene zum Einsatz. Mit der generischen Umsetzung von μDTN hingegen lassen sich diverse Szenarien mit ein und derselben Implementierung zufriedenstellend betreiben.

Auch eine Datenreduktion mittels einer angepassten Deltakodierung hat das Potential, in anderen Bereichen, bei denen zeitlich voneinander abhängige Daten aufgenommen werden, mit wenig Aufwand eine große Ersparnis zu bewirken: Solange die aufzunehmenden Daten nicht vollständig zufällig verteilt sind, kann eine an den Anwendungsfall angepasste Kodierung mit wenig Rechenaufwand eine größere Ersparnis bringen als etablierte (und vermeintlich optimale) Verfahren, die außerdem mehr Ressourcen benötigen.

Die Verschlüsselung mit OTP ist an sich zwar keine neue Idee, ihre Anwendung im Bereich von Gesundheitspflege-, Notfall- und AAL-Szenarien allerdings schon. Da diese Verschlüsselung nur wenig Rechenressourcen benötigt und dabei ein hohes Maß an Sicherheit bietet, kann diese Technik auch in vielen anderen Bereichen der drahtlosen Sensornetze Anwendung finden, in denen verhältnismäßig wenige (aber schützenswerte) Daten anfallen. Dazu müssen zukünftige Anwendungsfälle entweder initial mit einer ausreichend großen Menge an Schlüsseln ausgestattet sein, oder es müssen geeignete Mechanismen zur Schlüsselverteilung (beispielsweise durch das Austauschen von SD-Karten) berücksichtigt werden. Auch hier könnten die im Rahmen dieser Dissertation vorgestellten Arbeiten eingesetzt werden.

Insgesamt können die vorgestellten Ansätze und Lösungen also auch über die behandelten Szenarien hinaus eine breite Anwendung finden.

Glossar

AAL Ambient Assisted Living – Umgebungsunterstütztes Leben ... iii

ADC Analog-to-Digital Converter – Analog-digital-Wandler ... 21

ADL Activities of daily living – Aktivitäten des täglichen Lebens ... 5

AD Analog-Digital ... 22

AES Advanced Encryption Standard – Symmetrisches Verschlüsselungsverfahren (Blockchiffre) ... 79

API Application Programming Interface – Anwendungsprogrammierschnittstelle ... 67

BAN Body Area Network – Netzwerk aus am Körper getragenen Sensoren ... 6

BP Bundle-Protokoll ... 70

CA Certificate Authority – Zertifizierungsstelle ... 81

CBC Cipher Block Chaining – Betriebsart für Blockchiffren ... 80

CBHE Compressed Bundle Header Encoding ... 66

CFB Cipher Feedback – Betriebsart für Blockchiffren ... 80

CL Convergence Layer ... 69

CTR Counter – Betriebsart für Blockchiffren ... 80

DSP Digitaler Signalprozessor ... 99

DTN Disruption Tolerant Network – Unterbrechungs- bzw. verzögerungstolerantes Netzwerk ... 8

ECB Electronic Code Book – Betriebsart für Blockchiffren ... 79

ECC Elliptic Curve Cryptography ... 79

EEG Elektroenzephalografie ... 31

EID Endpoint Identifier – Eindeutiger Bezeichner ... 66

EKG Elektrokardiogramm ... 6

EMG Elektromyografie ... 6

FAT File Allocation Table – Universelles Dateisystem ... 92

FIFO First In–First Out – Speicherverfahren, welches die Elemente in Eingangsreihenfolge ausgibt ... 48

GAL Niedersächsischen Forschungsverbunds zur Gestaltung altersgerechter Lebenswelten – Informations- und Kommunikationstechnik zur Gewinnung und Aufrechterhaltung von Lebensqualität, Gesundheit und Selbstbestimmung in der zweiten Lebenshälfte ... 3

HTTP HyperText Transfer Protocol – Im WWW eingesetztes Protokoll ... 26

HTTPS HyperText Transfer Protocol Secure – Mit TLS gesichertes HTTP ... 86

I^2C Inter-Integrated Circuit – Bus zur Verbindung von Bauteilen ... 21

IKE Internet Key Exchange ... 86

INGA Inexpensive Node for General Applications – Ein kostengünstiger drahtloser Sensorknoten für universelle Anwendungen ... 41

IPND IP Neighbor Discovery ... 67

IPsec Internet Protocol Security – Sicherheitsmechanismen für das IP ... 86

L2TP Layer 2 Transport Protocol – Tunnel-Protokoll auf Ebene der Sicherungsschicht ... 146

LQI Link Quality Indicator ... 74

LTE Long Term Evolution – Mobilfunknetze der vierten Generation ... 150

MAC Medium Access Control – Medienzugriffssteuerung ... 21

MAC Message Authenticity Code – Code zur Authentifizierung ... 85

MEMS Microelectromechanical systems ... 10

MFR MAC-Footer ... 69

MHR MAC-Header ... 68

μDTN μDTN – Unterbrechungstolerantes Übertragungsprotokoll für Sensorknoten ... 8

MSHP Multi-Services Home Platform – Hard- und Softwareplattform im GAL-Projekt ... 5

NAT Network Address Translation ... 138

NIST National Institute of Standards and Technology ... 79

OFB Cipher Feedback – Betriebsart für Blockchiffren ... 80

OTP One-Time-Pad – Schlüssel, die nur ein Mal benutzt werden ... 8

PC Personal Computer ... 26

PERS Personal Emergency Response System – Hausnotrufsystem ... 12

PHY Physical Layer – Bitübertragungsschicht ... 68

PPTP Point-to-point Tunneling Protocol – Veraltetes Tunnel-Protokoll das auf IP aufsetzt ... 146

PSDU PHY Service Data Unit ... 68

RAM Random-Access Memory – Arbeitsspeicher ... 110

RSSI Received Signal Strength Indication ... 74

RTC Real Time Clock – Echtzeituhr ... 22

SDNV Self-Delimiting Numeric Value ... 70

SNMP Simple Network Management Protocol ... 141

SPI Serial Peripheral Interface – Bus zur Verbindung von Bauteilen ... 21

SPN Substitutions-Permutations-Netzwerk ... 79

SSL Secure Sockets Layer ... 86

SoC System-on-a-Chip ... 23

TCP Transmission Control Protocol – Transportprotokoll im Internet ... 86

TDMA Time Division Multiplex – Multiplexverfahren basierend auf (festen) Zeitschlitzen ... 21

TLS Transport Layer Security – Sicherheitsmechanismen, die auf TCP aufsetzen ... 86

UART Universal Asynchronous Receiver Transmitter ... 49

UDP User Datagram Protocol – Transportprotokoll im Internet ... 70

Literaturverzeichnis

[1] Statistisches Bundesamt. Alleinlebende in Deutschland – Ergebnisse des Mikrozensus 2011. Wiesbaden, 2012.

[2] Reinhold Haux, Andreas Hein, Marco Eichelberg, Jens-E. Appell, Hans-Jürgen Appelrath, Christian Bartsch, Thomas Bisitz, Jörg Bitzer, Matthias Blau, Susanne Boll, Michael Buschermöhle, Felix Büsching, Birte Erdmann, Uwe Fachinger, Juliane Felber, Tobias Fleuren, Matthias Gietzelt, Stefan Goetze, Mehmet Gövercin, Axel Helmer, Wilko Heuten, Volker Hohmann, Rainer Huber, Manfred Hülsken-Giesler, Gerold Jacobs, Riana Kayser, Arno Kerling, Timo Klingeberg, Harald Künemund Yvonne Költzsch, Jennifer Kunze, Wolfram Ludwig, Michael Marschollek, Birger Martens, Markus Meis, Eike Michael Meyer, Jochen Meyer, Wolfgang Nebel, Franz J. Neyer, Petra-Karin Okken, Hartmut Remmers, Lars Rölker-Denker, Thomas Rohdenburg, Meinhard Schilling, Giesela C. Schulze, Bianying Song, Jens Spehr, Elisabeth Steinhagen-Thiessen, Uwe Tegtbur, Wilfried Thoben, Peter Van Hengel, Stefan Wabnik, Friedrich Wahl, Sandra Wegel, Olaf Wilken, Simon Winkelbach, Thorben Wist, Manfred Wittrock, Klaus-Hendrik Wolf, Lars Wolf, and Melanie Zokoll-Van DerLaan. The Lower Saxony research network design of environments for ageing: towards interdisciplinary research on information and communication technologies in ageing societies. *Informatics for Health and Social Care*, 35(3-4):92–103, 12 2010.

[3] Andreas Hein, Simon Winkelbach Birger Martens, Olaf Wilken, Marco Eichelberg, Jens Spehr, Matthias Gietzelt, Klaus-Hendrik Wolf, Felix Büsching, Manfred Hülsken-Giesler, Markus Meis, and Petra Okken. Monitoring systems for the support of home care. *Informatics for Health and Social Care*, 35(3-4):157–176, 12 2010.

[4] Peter Stok, Holger Karl, Fabio Ambroggi, Jean-Dominique Decotignie, Frank Siegemund, Kees Lokhorst, and Michael Hellenschmidt. WASP – Wirelessly Accessible Sensor Populations: A Project Overview. In Max Mühlhäuser, Alois Ferscha, and Erwin Aitenbichler, editors, *Constructing Ambient Intelligence*, volume 11 of *Communications in Computer and Information Science*, pages 426–429. Springer Berlin Heidelberg, 2008.

[5] Martijn Bennebroek, Andre Barroso, Louis Atallah, Benny Lo, and Guang-Zhong Yang. Deployment of wireless sensors for remote elderly monitoring. In *e-Health Networking Applications and Services (Healthcom), 2010 12th IEEE International Conference on*, pages 1–5, 2010.

[6] Martijn Bennebroek, Junaid Ansari, Aleksandar Kovacevic, Xi Zhang, Elena Meshkova, and Petri Mähönen. Wirelessly Accessible Sensor Populations (WASP): Deployment and Enterprise Integration of Energy-Efficient Wireless Sensor Networks. In *Sensor, Mesh and Ad Hoc Communications and Networks Workshops, 2009. SECON Workshops '09. 6th Annual IEEE Communications Society Conference on*, pages 1–3, 2009.

[7] Louis Atallah, Benny Lo, Guang-Zhong Yang, and Frank Siegemund. Wirelessly accessible sensor populations (WASP) for elderly care monitoring. In *Pervasive Computing Technologies for Healthcare, 2008. PervasiveHealth 2008. Second International Conference on*, pages 2–7, 2008.

[8] David Malan, Thaddeus Fulford-jones, Matt Welsh, and Steve Moulton. CodeBlue: An ad hoc sensor network infrastructure for emergency medical care. In *In International Workshop on Wearable and Implantable Body Sensor Networks*, 2004.

[9] Victor Shnayder, Bor-rong Chen, Konrad Lorincz, Thaddeus R. F. Fulford Jones, and Matt Welsh. Sensor networks for medical care. In *Proceedings of the 3rd international conference on Embedded networked sensor systems*, SenSys '05, pages 314–314, New York, NY, USA, 2005. ACM.

[10] Tia Gao, Christopher Pesto, Leo Selavo, Yin Chen, JeongGil Ko, Jong Hyun Lim, Andreas Terzis, Andrew Watt, James Jeng, Bor rong Chen, Konrad Lorincz, and Matt Welsh. Wireless Medical Sensor Networks in Emergency Response: Implementation and Pilot Results. In *Technologies for Homeland Security, 2008 IEEE Conference on*, pages 187–192, 2008.

[11] JeongGil Ko, Tia Gao, and Andreas Terzis. Empirical study of a medical sensor application in an urban emergency department. In *Proceedings of the Fourth International Conference on Body Area Networks*, BodyNets '09, pages 10:1–10:8, ICST, Brussels, Belgium, Belgium, 2009. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[12] Andreas Schrader, Darren Carlson, and Peter Rothenpieler. SmartAssist - Wireless Sensor Networks for Unobtrusive Health Monitoring. In *Proceedings of the 5th BMI Workshop on Behaviour Monitoring and Interpretation*, volume 678, pages 84–89, Karlsruhe, Germany, 2010.

[13] Caroline Rougier, Jean Meunier, Alain St-Arnaud, and Jacqueline Rousseau. Fall Detection from Human Shape and Motion History Using Video Surveillance. In *Advanced Information Networking and Applications Workshops, 2007, AINAW '07. 21st International Conference on*, volume 2, pages 875–880, 2007.

[14] Andrew Sixsmith, N. Johnson, and Roger W. Whatmore. Pyroelectric IR sensor arrays for fall detection in the older population. *J. Phys. IV France*, 128:153–160, 2005.

[15] Tomàs Pallejà, Mercè Teixidó, Marcel Tresanchez, and Jordi Palacín. Measuring gait using a ground laser range sensor. *Sensors*, 9(11):9133–9146, 2009.

[16] Yaniv Zigel, Dima Litvak, and Israel Gannot. A Method for Automatic Fall Detection of Elderly People Using Floor Vibrations and Sound - Proof of Concept on Human Mimicking Doll Falls. *Biomedical Engineering, IEEE Transactions on*, 56(12):2858–2867, 2009.

[17] JeongGil Ko, Chenyang Lu, Mani B. Srivastava, John A. Stankovic, Andreas Terzis, and Matt Welsh. Wireless Sensor Networks for Healthcare. *Proceedings of the IEEE*, 98(11):1947–1960, 2010.

[18] Maarit Kangas, Antti Konttila, Per Lindgren, Ilkka Winblad, and Timo Jämsä. Comparison of low-complexity fall detection algorithms for body attached accelerometers . *Gait & Posture*, 28(2):285 – 291, 2008.

[19] Norbert Noury, Anthony Fleury, Pierre Rumeau, Alan K. Bourke, Gearoid O. Laighin, Vincent Rialle, and Jean-Eric Lundy. Fall detection - Principles and Methods. In *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE*, pages 1663–1666, 2007.

[20] Susan L. Murphy. Review of physical activity measurement using accelerometers in older adults: Considerations for research design and conduct. *Preventive Medicine*, 48(2):108 – 114, 2009.

[21] J. Chen, Karric Kwong, Dennis Chang, Jerry Luk, and Ruzena Bajcsy. Wearable Sensors for Reliable Fall Detection. In *Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005. 27th Annual International Conference of the*, pages 3551 –3554, 01 2005.

[22] Alan K. Bourke and Gerard M. Lyons. A threshold-based fall-detection algorithm using a bi-axial gyroscope sensor. *Medical Engineering & Physics*, 30(1):84 – 90, 2008.

[23] Qiang Li, John A. Stankovic, Mark A. Hanson, Adam T. Barth, John Lach, and Gang Zhou. Accurate, Fast Fall Detection Using Gyroscopes and Accelerometer-Derived Posture Information. In *Wearable and Implantable Body Sensor Networks, 2009. BSN 2009. Sixth International Workshop on*, pages 138–143, 2009.

[24] Federico Bianchi, Stephen J. Redmond, Michael R. Narayanan, Sergio Cerutti, Branko G. Celler, and Nigel H. Lovell. Falls event detection using triaxial accelerometry and barometric pressure measurement. In *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, pages 6111 –6114, sept. 2009.

[25] Jiangpeng Dai, Xiaole Bai, Zhimin Yang, Zhaohui Shen, and Dong Xuan. Mobile phone-based pervasive fall detection. *Personal and Ubiquitous Computing*, 14(7):633–643, 2010.

[26] Frank Sposaro and Gary Tyson. iFall: An android application for fall monitoring and response. In *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, pages 6119–6122, 2009.

[27] Diane Podsiadlo and Sandra Richardson. The timed „Up & Go“: a test of basic functional mobility for frail elderly persons. *Journal of the American Geriatrics Society*, 39(2):142–148, 1991.

[28] Tobias Baumgartner, Sándor P. Fekete, Tom Kamphans, Alexander Kröller, Max Pagel, Matthias Gietzelt, and Reinhold Haux. Demo Abstract: Using a Sensor Network to Enhance a Standardized Medical Test. In Pedro José Marrón and Kamin Whitehouse, editors, *EWSN*, volume 6567 of *Lecture Notes in Computer Science*. Springer, 2011.

[29] Michael Marschollek, Klaus-Hendrik Wolf, Matthias Gietzelt, Gerhard Nemitz, Hubertus Meyer zu Schwabedissen, and Reinhold Haux. Assessing elderly persons' fall risk using spectral analysis on accelerometric data - a clinical evaluation study. In *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*, pages 3682 –3685, august 2008.

[30] Barry R. Greene, Denise McGrath, Karol J. O'Donovan, Ross O'Neill, Adrian Burns, and Brian Caulfield. Adaptive estimation of temporal gait parameters using body-worn gyroscopes. In *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pages 1296 –1299, 31 2010-sept. 4 2010.

[31] Ion P.I. Pappas, Thierry Keller, Sabine Mangold, Milos R. Popovic, Volker Dietz, and Manfred Morari. A reliable gyroscope-based gait-phase detection sensor embedded in a shoe insole. *Sensors Journal, IEEE*, 4(2):268 – 274, april 2004.

[32] Ruth E. Mayagoitia, Anand V. Nene, and Peter H. Veltink. Accelerometer and rate gyroscope measurement of kinematics: an inexpensive alternative to optical motion analysis systems. *Journal of Biomechanics*, 35(4):537 – 542, 2002.

[33] Ling Bao and Stephen S. Intille. Activity Recognition from User-Annotated Acceleration Data. In Alois Ferscha and Friedemann Mattern, editors, *Pervasive Computing*, volume 3001 of *Lecture Notes in Computer Science*, pages 1–17. Springer Berlin Heidelberg, 2004.

[34] Miikka Ermes, Juha Pärkkä, Jani Mäntyjärvi, and Ilkka Korhonen. Detection of Daily Activities and Sports With Wearable Sensors in Controlled and Uncontrolled Conditions. *Information Technology in Biomedicine, IEEE Transactions on*, 12(1):20 –26, jan. 2008.

[35] Louis Atallah, Benny Lo, Rachel King, and Guang-Zhong Yang. Sensor placement for activity detection using wearable accelerometers. In *Body Sensor Networks (BSN), 2010 International Conference on*, pages 24–29. IEEE, 2010.

[36] Jonathan Lester, Tanzeem Choudhury, Nicky Kern, Gaetano Borriello, and Blake Hannaford. A hybrid discriminative/generative approach for modeling human activities. In *Proceedings of the 19th international joint conference on Artificial intelligence*, IJCAI'05, pages 766–772, San Francisco, CA, USA, 2005. Morgan Kaufmann Publishers Inc.

[37] Gerhard Lammel, Johannes Gutmann, Lukas M. Marti, and M. Dobler. Indoor Navigation with MEMS sensors. *Procedia Chemistry*, 1(1):532 – 535, 2009.

[38] Alex Moschevikin, Roman Voronov, Alexandr Galov, and Alexei Soloviev. Using pressure sensors for floor identification in wireless sensors networks. In *Wireless Systems (IDAACS-SWS), 2012 IEEE 1st International Symposium on*, pages 2–6, 2012.

[39] Melanie Swan. Emerging Patient-Driven Health Care Models: An Examination of Health Social Networks, Consumer Personalized Medicine and Quantified Self-Tracking. *International Journal of Environmental Research and Public Health*, 6(2):492–525, 2009.

[40] Peter Wagner and Victor Shkawrytko. Personal alarm apparatus including wrist supported transmitter and receiver/telephone interface circuit, April 9 1985. US Patent 4,510,350.

[41] Felix Büsching, Henning Post, Matthias Gietzelt, and Lars Wolf. Fall Detection on the Road. In *2013 IEEE 15th International Conference on e-Health Networking, Applications and Services (Healthcom) (IEEE Healthcom 2013)*, Lisbon, Portugal, October 2013.

[42] Bundesamt für Sicherheit in der Informationstechnik (BSI), editor. *Informationssicherheit und IT-Grundschutz: BSI-Standards 100-1, 100-2 und 100-3*. Bundesanzeiger, 06 2008.

[43] Peter Rothenpieler, Claudia Becker, and Stefan Fischer. Privacy Concerns in a Remote Monitoring and Social Networking Platform for Assisted Living. In Simone Fischer-Hübner, Penny Duquenoy, Marit Hansen, Ronald Leenes, and Ge Zhang, editors, *Privacy and Identity Management for Life*, volume 352 of *IFIP Advances in Information and Communication Technology*, pages 219–230. Springer Berlin Heidelberg, 2011.

[44] Bruce Schneier. Why cryptography is harder than it looks. In *Edi Forum -OAK PARK-*, volume 10, pages 87–90. The Edi Group, LTD., 1997.

[45] Tony O'Donovan, James Brown, Felix Büsching, Alberto Cardoso, Jose Cecelio, Jose do O, Pedro Furtado, Paulo Gil, Anja Jugel, Wolf-Bastian Pöttner, Utz Roedig, Jorge sa Silva, Ricardo Silva, Cormac J. Sreenan, Vasos Vassiliou, Thiemo Voigt, Lars Wolf, and Zinon Zinonos. The GINSENG System for Wireless Monitoring and Control: Design and Deployment Experiences. *ACM Trans. Sen. Netw.*, 10(1):4:1–4:40, November 2013.

[46] Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li Shiuan Peh, and Daniel Rubenstein. Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with ZebraNet. In *Proceedings of the 10th international conference on Architectural support for programming languages and operating systems*, ASPLOS-X, pages 96–107, New York, NY, USA, 2002. ACM.

[47] Sukun Kim, Shamim Pakzad, David Culler, James Demmel, Gregory Fenves, Steve Glaser, and Martin Turon. Wireless sensor networks for structural health monitoring. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, SenSys '06, pages 427–428, New York, NY, USA, 2006. ACM.

[48] Prabal Dutta and David Culler. Epic: An Open Mote Platform for Application-Driven Design. In *Proceedings of the 7th international conference on Information processing in sensor networks*, IPSN '08, pages 547–548, Washington, DC, USA, 2008. IEEE Computer Society.

[49] Jochen H. Schiller, Achim Liers, and Hartmut Ritter. ScatterWeb: A wireless sensornet platform for research and teaching. *Computer Communications*, 28(13):1545–1551, 2005.

[50] Rahul Mangharam, Anthony Rowe, and Raj Rajkumar. FireFly: a cross-layer platform for real-time embedded wireless networks. *Real-Time Systems*, 37:183–231, 2007. 10.1007/s11241-007-9028-z.

[51] Konrad Lorincz, Bor-rong Chen, Geoffrey Werner Challen, Atanu Roy Chowdhury, Shyamal Patel, Paolo Bonato, and Matt Welsh. Mercury: a wearable sensor network platform for high-fidelity motion analysis. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, SenSys '09, pages 183–196, New York, NY, USA, 2009. ACM.

[52] Joseph Polastre, Robert Szewczyk, and David Culler. Telos: enabling ultra-low power wireless research. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, page 48, Piscataway, NJ, USA, 2005. IEEE Press.

[53] Adam Dunkels, Björn Grönvall, and Thiemo Voigt. Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors. In *Proceedings of the First IEEE Workshop on Embedded Networked Sensors (Emnets-I)*, Tampa, Florida, USA, November 2004.

[54] TinyOS Alliance. TinyOS 2.1 adding threads and memory protection to TinyOS. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, SenSys '08, pages 413–414, New York, NY, USA, 2008. ACM.

[55] Tony O'Donovan, James Brown, Utz Roedig, Cormac J. Sreenan, Jose do O, Adam Dunkels, Anja Klein, Jorge Sa Silva, Vasos Vassiliou, and Lars Wolf. GINSENG: Performance Control in Wireless Sensor Networks. In *Sensor Mesh and Ad Hoc Communications and Networks (SECON), 2010 7th Annual IEEE Communications Society Conference on*, pages 1 –3, june 2010.

[56] Ioannis Chatzigiannakis, Stefan Fischer, Christos Koninis, Georgios Mylonas, and Dennis Pfisterer. WISEBED: An Open Large-Scale Wireless Sensor Network Testbed. In Nikos Komninos, editor, *Sensor Applications, Experimentation, and Logistics*, volume 29 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 68–87. Springer Berlin Heidelberg, 2010.

[57] Tobias Baumgartner, SándorP. Fekete, Tom Kamphans, Alexander Kröller, and Max Pagel. Hallway Monitoring: Distributed Data Processing with Wireless Sensor Networks. In PedroJ. Marron, Thiemo Voigt, Peter Corke, and Luca Mottola, editors, *Real-World Wireless Sensor Networks*, volume 6511 of *Lecture Notes in Computer Science*, pages 94–105. Springer Berlin Heidelberg, 2010.

[58] Gregory J. Welk, James J. McClain, Joey C. Eisenmann, and Eric E. Wickel. Field Validation of the MTI Actigraph and BodyMedia Armband Monitor Using the IDEEA Monitor. *Obesity*, 15(4):918–928, 2007.

[59] Girardin Jean-Louis, Daniel F Kripke, Roger J Cole, Joseph D Assmus, and Robert D Langer. Sleep detection with an accelerometer actigraph: comparisons with polysomnography. *Physiology & Behavior*, 72(1–2):21 – 28, 2001.

[60] Warren W. Tryon, Georgiana Shick Tryon, Thomas Kazlausky, William Gruen, and James M. Swanson. Reducing Hyperactivity with a Feedback Actigraph: Initial Findings. *Clinical Child Psychology and Psychiatry*, 11(4):607–617, 2006.

[61] Daniel Arvidsson, Frode Slinde, Sven Larsson, and Lena Hulthen. Energy cost of physical activities in children : Validation of SenseWear Armband. *Medicine & Science in Sports & Exercise*, 39(11):2076–2084, 2007.

[62] Michael Adelmann. *Vergleichende Untersuchung über das Ausmaß der Alltagsaktivität bei Krankenpflegerinnen und Sekretärinnen mittels des SenseWear-Armbandes*. GRIN Verlag, 2007.

[63] J. Postel. Transmission Control Protocol. RFC 793 (INTERNET STANDARD), September 1981. Updated by RFCs 1122, 3168, 6093, 6528.

[64] J. Postel. Internet Protocol. RFC 791 (INTERNET STANDARD), September 1981. Updated by RFCs 1349, 2474.

[65] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Draft Standard), June 1999. Updated by RFCs 2817, 5785, 6266, 6585.

[66] Adam Dunkels, Thiemo Voigt, and Juan Alonso. Making TCP/IP Viable for Wireless Sensor Networks. In *Proceedings of the First European Workshop on Wireless Sensor Networks (EWSN 2004), work-in-progress session*, Berlin, Germany, January 2004.

[67] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard), December 1998. Updated by RFCs 5095, 5722, 5871, 6437, 6564.

[68] Geoff Mulligan. The 6LoWPAN architecture. In *Proceedings of the 4th workshop on Embedded networked sensors*, EmNets '07, pages 78–82, New York, NY, USA, 2007. ACM.

[69] Chen Yibo, Kun mean Hou, Haiying Zhou, Hong-Ling Shi, Xing Liu, Xunxing Diao, Hao Ding, Jian-Jin Li, and C. de Vaulx. 6LoWPAN Stacks: A Survey. In *Wireless Communications, Networking and Mobile Computing (WiCOM), 2011 7th International Conference on*, pages 1–4, 2011.

[70] Adam Dunkels, Fredrik Österlind, and Zhitao He. An adaptive communication architecture for wireless sensor networks. In *Proceedings of the 5th international conference on Embedded networked sensor systems*, SenSys '07, pages 335–349, New York, NY, USA, 2007. ACM.

[71] Marco Eichelberg, Andres Hein, Felix Büsching, and Lars Wolf. The GAL Middleware Platform for AAL. In *Proceedings of the First International Workshop on AAL Service Platforms (WASP 2010)*, Lyon, France, July 2010.

[72] Evangelos Bekiaris and Silvio Bonfiglio. The OASIS Concept. In Constantine Stephanidis, editor, *Universal Access in Human-Computer Interaction. Addressing Diversity*, volume 5614 of *Lecture Notes in Computer Science*, pages 202–209. Springer Berlin Heidelberg, 2009.

[73] Mohammad-Reza Tazari, Francesco Furfari, Juan-PabloLázaro Ramos, and Erina Ferro. The PERSONA Service Platform for AAL Spaces. In Hideyuki Nakashima, Hamid Aghajan, and JuanCarlos Augusto, editors, *Handbook of Ambient Intelligence and Smart Environments*, pages 1171–1199. Springer US, 2010.

[74] Andrew Sixsmith, Sonja Meuller, Felicitas Lull, Michael Klein, Ilse Bierhoff, Sarah Delaney, and Robert Savage. SOPRANO – An Ambient Assisted Living System for Supporting Older People at Home. In

Mounir Mokhtari, Ismail Khalil, Jérémy Bauchet, Daqing Zhang, and Chris Nugent, editors, *Ambient Assistive Health and Wellness Management in the Heart of the City*, volume 5597 of *Lecture Notes in Computer Science*, pages 233–236. Springer Berlin Heidelberg, 2009.

[75] Felix Büsching, Michael Doering, and Lars Wolf. Integration of an Environments for Aging Platform in SOHO-Routers. In *Proceedings of the 14th IEEE International Symposium on Consumer Electronics (ISCE2010)*, Braunschweig, Germany, June 2010.

[76] Axel Helmer, Marco Eichelberg, Markus Meis, Matthias Gietzelt, Olaf Wilken, and Andreas Hein. Ein System zur eskalierenden Notruf- und Informationsweiterleitung vom Heimumfeld älterer Menschen zu externen Anwendern. In *Tagungsband Ambient Assisted Living*, Berlin, Offenbach, Bismarckstraße 33, 10625 Berlin, 01 2010. VDE, VDE VERLAG GMBH.

[77] Claude Elwood Shannon. Communication in the presence of noise. *Proceedings of the IEEE*, 72(9):1192–1201, 1984.

[78] Bernard Widrow. Statistical analysis of amplitude-quantized sampled-data systems. *American Institute of Electrical Engineers, Part II: Applications and Industry, Transactions of the*, 79(6):555–568, 1961.

[79] Felix Büsching, Ulf Kulau, Matthias Gietzelt, and Lars Wolf. Comparison and Validation of Capacitive Accelerometers for Health Care Applications. *Computer Methods and Programs in Biomedicine*, 106(2):79 – 88, 2012.

[80] Antoon Th.M. Willemsen, Jan A. van Alsté, and Herman B.K. Boom. Real-time gait assessment utilizing a new way of accelerometry. *Journal of Biomechanics*, 23(8):859 – 863, 1990.

[81] Kamiar Aminian, Bijan Najafi, Christophe Büla, Pierre-François Leyvraz, and Ph Alain Robert. Spatio-temporal parameters of gait measured by an ambulatory system using miniature gyroscopes. *Journal of Biomechanics*, 35(5):689 – 699, 2002.

[82] Tao Liu, Yoshio Inoue, and Kyoko Shibata. Development of a wearable sensor system for quantitative gait analysis. *Measurement*, 42(7):978 – 988, 2009.

[83] Sergio Scapellato, Filippo Cavallo, Chiara Martelloni, and Angelo M. Sabatini. In-use calibration of body-mounted gyroscopes for applications in gait analysis. *Sensors and Actuators A: Physical*, 123–124(0):418 – 422, 2005.

[84] Federico Bianchi, Stephen J. Redmond, Michael R. Narayanan, Sergio Cerutti, and Nigel H. Lovell. Barometric Pressure and Triaxial Accelerometry-Based Falls Event Detection. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 18(6):619 –627, dec. 2010.

[85] Geoffrey P Dobson. On being the right size: heart design, mitochondrial efficiency and lifespan potential. *Clinical and Experimental Pharmacology and Physiology*, 30(8):590–597, 2003.

[86] Rune Fensli, Einar Gunnarson, and Ole Hejlesen. A wireless ECG system for continuous event recording and communication to a clinical alarm station. In *Engineering in Medicine and Biology Society, 2004. IEMBS '04. 26th Annual International Conference of the IEEE*, volume 1, pages 2208 –2211, sept. 2004.

[87] Joshua Proulx, Ryan Clifford, Sarah Sorensen, Dah-Jye Lee, and James Archibald. Development and Evaluation of a Bluetooth EKG Monitoring Sensor. In *Computer-Based Medical Systems, 2006. CBMS 2006. 19th IEEE International Symposium on*, pages 507–511, 2006.

[88] Bundesnetzagentur. Allgemeinzuteilung von Frequenzen in den Frequenzteilbereichen gemäß Frequenzbereichszuweisungsplanverordnung (FreqBZPV), Teil B: Nutzungsbestimmungen (NB) D138 und D150 für die Nutzung durch die Allgemeinheit für ISM-Anwendungen. Technical Report Vfg 76 / 2003, Bundesrepublik Deutschland, 2003.

[89] Claude Elwood Shannon. A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(1):3–55, January 2001. Reprint.

[90] IEEE Standards Association. IEEE Standard for Information technology –Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)*, pages 1–2793, 2012.

[91] IEEE Standards Association. IEEE Standard for Information Technology Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Specific Requirements. - Part 15.1: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs). *IEEE Std 802.15.1-2005 (Revision of IEEE Std 802.15.1-2002)*, pages 1–580, 2005.

[92] Armin Bolz and Claudius Moor. Bluetooth-Technologie als Kabelersatz im klinischen Umfeld. *Kardiotechnik*, 4:105–109, Dec 2007.

[93] IEEE Standards Association. IEEE Standard for Local and metropolitan area networks - Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 4: Alternative Physical Layer Extension to Support Medical Body Area Network (MBAN) Services Operating in the 2360 MHz 2400 MHz Band. *IEEE Std 802.15.4j-2013 (Amendment to IEEE Std 802.15.4-2011 as amended by IEEE Std 802.15.4e-2012, IEEE Std 802.15.4f-2012, and IEEE Std 802.15.4g-2012)*, pages 1–24, 2013.

[94] Patrick Kinney et al. Zigbee technology: Wireless control that simply works. In *Communications design conference*, volume 2, 2003.

[95] T. Ryan Burchfield, Subbarayan Venkatesan, and Douglas Weiner. Maximizing throughput in ZigBee wireless networks through analysis, simulations and implementations. In *Proceedings of the International Workshop on Localized Algorithms and Protocols for Wireless Sensor Networks (LOCALGOS 2007)*, pages 15–29, Santa Fe, New Mexico, June 2007. CTI Press, Athens.

[96] Fredrik Österlind and Adam Dunkels. Approaching the maximum 802.15.4 multihop throughput. In *In Proceedings of the Fifth ACM Workshop on Embedded Networked Sensors (HotEmNets 2008), June 2008*, 2008.

[97] Andrew R. Golding and Neal Lesh. Indoor navigation using a diverse set of cheap, wearable sensors. In *Wearable Computers, 1999. Digest of Papers. The Third International Symposium on*, pages 29–36, 1999.

[98] Farid Touati, Rohan Tabish, and Adel Ben Mnaouer. Towards u-health: An indoor 6LoWPAN based platform for real-time healthcare monitoring. In *Wireless and Mobile Networking Conference (WMNC), 2013 6th Joint IFIP*, pages 1–4, 2013.

[99] IEEE Standard Test Access Port and Boundary Scan Architecture. *IEEE Std 1149.1-2001*, pages 1–212, 2001.

[100] Tobias Baumgartner, Ioannis Chatzigiannakis, Sándor Fekete, Christos Koninis, Alexander Kröller, and Apostolos Pyrgelis. Wiselib: A Generic Algorithm Library for Heterogeneous Sensor Networks. In JorgeSá Silva, Bhaskar Krishnamachari, and Fernando Boavida, editors, *Wireless Sensor Networks*, volume 5970 of *Lecture Notes in Computer Science*, pages 162–177. Springer Berlin Heidelberg, 2010.

[101] Ulf Kulau, Felix Büsching, and Lars Wolf. A Node's Life: Increasing WSN Lifetime by Dynamic Voltage Scaling. In *The 9th IEEE International Conference on Distributed Computing in Sensor Systems 2013 (IEEE DCoSS 2013)*, Cambridge, USA, May 2013. accepted for publication.

[102] Felix Büsching and Lars Wolf. Chancen für den Einsatz von unterbrechungstoleranten Kommunikationsprotokollen in drahtlosen körpernahen Netzwerken im medizinischen Umfeld. In *Proceedings of the 42th Annual Conference of the Gesellschaft für Informatik e.V. (INFORMATIK 2012)*, volume 208, Braunschweig, Germany, Sep 2012. Köllen Druck + Verlag GmbH.

[103] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss. Delay-Tolerant Networking Architecture. RFC 4838 (Informational), April 2007.

[104] K. Scott and S. Burleigh. Bundle Protocol Specification. RFC 5050 (Experimental), November 2007.

[105] Sebastian Schildt, Johannes Morgenroth, Wolf-Bastian Pöttner, and Lars Wolf. IBR-DTN: A lightweight, modular and highly portable Bundle Protocol implementation. *Electronic Communications of the EASST*, 37:1–11, Jan 2011.

[106] Georg von Zengen, Felix Büsching, Wolf-Bastian Pöttner, and Lars Wolf. An Overview of μDTN: Unifying DTNs and WSNs. In *Proceedings of the 11th GI/ITG KuVS Fachgespräch Drahtlose Sensornetze (FGSN)*, Darmstadt, Germany, 9 2012.

[107] S. Burleigh. Compressed Bundle Header Encoding (CBHE). RFC 6260 (Experimental), May 2011.

[108] D. Ellard and D. Ellard. DTN IP Neighbor Discovery (IPND). *IRTF Draft*, 2010.

[109] D. McGrew and D. Bailey. AES-CCM Cipher Suites for Transport Layer Security (TLS). RFC 6655 (Proposed Standard), July 2012.

[110] Wolf-Bastian Pöttner, Felix Büsching, Georg von Zengen, and Lars Wolf. Data Elevators: Applying the Bundle Protocol in Delay Tolerant Wireless Sensor Networks. In *The Ninth IEEE International Conference on Mobile Ad-hoc and Sensor Systems (IEEE MASS 2012)*, Las Vegas, Nevada, USA, October 2012.

[111] Felix Büsching, Maximiliano Bottazzi, Wolf-Bastian Pöttner, and Lars Wolf. DT-WBAN: Disruption Tolerant Wireless Body Area Networks in Healthcare Applications. In *the International Workshop on e-Health Pervasive Wireless Applications and Services (eHPWAS'13)*, Lyon, France, Oct 2013. Best Paper Award.

[112] Ad Kamerman and Guido Aben. Throughput performance of wireless LANs operating at 2.4 and 5 GHz. In *Personal, Indoor and Mobile Radio Communications, 2000. PIMRC 2000. The 11th IEEE International Symposium on*, volume 1, pages 190–195 vol.1, 2000.

[113] Yang Xiao. IEEE 802.11n: enhancements for higher throughput in wireless LANs. *Wireless Communications, IEEE*, 12(6):82–91, 2005.

[114] Adrian Perrig, John Stankovic, and David Wagner. Security in Wireless Sensor Networks. *Commun. ACM*, 47(6):53–57, June 2004.

[115] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978.

[116] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. Cryptographic communications system and method, September 1983.

[117] F. Amin, A.H. Jahangir, and H. Rasifard. Analysis of public-key cryptography for wireless sensor networks security. *World Academy of Science, Engineering and Technology*, 41:529–534, 2008.

[118] Abdullah Said Alkalbani, Teddy Mantoro, and Abu Osman Md Tap. Comparison between RSA hardware and software implementation for WSNs security schemes. In *Information and Communication Technology for the Muslim World (ICT4M), 2010 International Conference on*, pages E84 –E89, dec. 2010.

[119] Vipul Gupta, Matthew Millard, Stephen Fung, Yu Zhu, Nils Gura, Hans Eberle, and Sheueling Chang Shantz. Sizzle: a standards-based end-to-end security architecture for the embedded Internet. In *Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on*, pages 247–256, 2005.

[120] Wooyoung Jung, Sungmin Hong, Minkeun Ha, Young-Joo Kim, and Daeyoung Kim. SSL-Based Lightweight Security of IP-Based Wireless Sensor Networks. In *Advanced Information Networking and Applications Workshops, 2009. WAINA '09. International Conference on*, pages 1112–1117, 2009.

[121] Yee Wei Law, Jeroen Doumen, and Pieter Hartel. Survey and benchmark of block ciphers for wireless sensor networks. *ACM Trans. Sen. Netw.*, 2(1):65–93, February 2006.

[122] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES–the Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.

[123] Eli Biham, Alex Biryukov, Orr Dunkelman, Eran Richardson, and Adi Shamir. Initial Observations on Skipjack: Cryptanalysis of Skipjack-3XOR. In Stafford Tavares and Henk Meijer, editors, *Selected Areas in Cryptography*, Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1999.

[124] Chris Karlof, Naveen Sastry, and David Wagner. TinySec: a link layer security architecture for wireless sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, SenSys '04, pages 162–175, New York, NY, USA, 2004. ACM.

[125] Mark Luk, Ghita Mezzour, Adrian Perrig, and Virgil Gligor. MiniSec: a secure sensor network communication architecture. In *Proceedings of the 6th international conference on Information processing in sensor networks*, IPSN '07, pages 479–488, New York, NY, USA, 2007. ACM.

[126] Roberto Doriguzzi Corin, Giovanni Russello, and Elio Salvadori. TinyKey: A light-weight architecture for Wireless Sensor Networks securing real-world applications. In *Wireless On-Demand Network Systems and Services (WONS), 2011 Eighth International Conference on*, jan. 2011.

[127] Lander Casado and Philippas Tsigas. ContikiSec: A Secure Network Layer for Wireless Sensor Networks under the Contiki Operating System. In *Proceedings of the 14th Nordic Conference on Secure IT Systems: Identity and Privacy in the Internet Age*, NordSec '09, Berlin, Heidelberg, 2009.

[128] Bruce Schneier. *Applied cryptography: protocols, algorithms, and source code in C*. Wiley, 1996.

[129] Anne Marie Hegland, Eli Winjum, Stig Fr Mjlsnes, Chunming Rong, ivind Kure, and Pål Spilling. A survey of key management in ad hoc networks. *IEEE Communications Surveys and Tutorials*, 8(1-4):48–66, 2006.

[130] Laurent Eschenauer and Virgil D. Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM conference on Computer and communications security*, CCS '02, pages 41–47, New York, NY, USA, 2002. ACM.

[131] Roberto Di Pietro, Luigi V. Mancini, and Alessandro Mei. Random key-assignment for secure Wireless Sensor Networks. In *Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks*, SASN '03, pages 62–71, New York, NY, USA, 2003. ACM.

[132] Haowen Chan and Adrian Perrig. PIKE: peer intermediaries for key establishment in sensor networks. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 1, pages 524–535 vol. 1, 2005.

[133] Li Zhou, Jinfeng Ni, and Chinya V. Ravishankar. Supporting Secure Communication and Data Collection in Mobile Sensor Networks. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–12, 2006.

[134] Shammi Didla, Aaron Ault, and Saurabh Bagchi. Optimizing AES for embedded devices and wireless sensor networks. In *Proceedings of the 4th International Conference on Testbeds and research infrastructures for the development of networks & communities*, TridentCom '08, ICST, Brussels, Belgium, Belgium, 2008.

[135] Shahid Raza, Simon Duquennoy, Antony Chung, Dogan Yazar, Thiemo Voigt, and Utz Roedig. Securing communication in 6LoWPAN with compressed IPsec. In *Distributed Computing in Sensor Systems and Workshops (DCOSS), 2011 International Conference on*, pages 1–8, 2011.

[136] S. Symington, S. Farrell, H. Weiss, and P. Lovell. Bundle Security Protocol Specification. RFC 6257 (Experimental), May 2011.

[137] Claude Elwood Shannon. Communication theory of secrecy systems. *Bell system technical journal*, 28(4):656–715, 1949.

[138] Ian Goldberg and David Wagner. Randomness and the netscape browser. *Dr Dobb's Journal-Software Tools for the Professional Programmer*, 21(1):66–71, 1996.

[139] Aurelien Francillon and Claude Castelluccia. TinyRNG: A Cryptographic Random Number Generator for Wireless Sensors Network Nodes. In *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks and Workshops, 2007. WiOpt 2007. 5th International Symposium on*, pages 1–7, 2007.

[140] Sebastian Schildt, Wolf-Bastian Pöttner, Felix Büsching, and Lars C Wolf. RATFAT: ReAl-Time FAT for Cooperative Multitasking Environments in WSNs. In *5th Workshop on Performance Control in Wireless Sensor Networks 2013 (PWSN 2013)*, Cambridge, USA, May 2013.

[141] Joel J.P.C. Rodrigues, João Caldeira, and Binod Vaidya. A Novel Intra-body Sensor for Vaginal Temperature Monitoring. *Sensors*, 9(4):2797–2808, 2009.

[142] David Albert Huffman. A Method for the Construction of Minimum-Redundancy Codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[143] Andreas Reinhardt, Delphine Christin, Matthias Hollick, Johannes Schmitt, ParagS. Mogre, and Ralf Steinmetz. Trimming the Tree: Tailoring Adaptive Huffman Coding to Wireless Sensor Networks. In JorgeSá Silva, Bhaskar Krishnamachari, and Fernando Boavida, editors, *Wireless Sensor Networks*, volume 5970 of *Lecture Notes in Computer Science*, pages 33–48. Springer Berlin Heidelberg, 2010.

[144] Alexandre Guitton, Niki Trigoni, and Sven Helmer. Fault-Tolerant Compression Algorithms for Delay-Sensitive Sensor Networks with Unreliable Links. In SotirisE. Nikoletseas, BogdanS. Chlebus, DavidB. Johnson, and Bhaskar Krishnamachari, editors, *Distributed Computing in Sensor Systems*, volume 5067 of *Lecture Notes in Computer Science*, pages 190–203. Springer Berlin Heidelberg, 2008.

[145] Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *Information Theory, IEEE Transactions on*, 23(3):337–343, 1977.

[146] Terry A. Welch. A Technique for High-Performance Data Compression. *Computer*, 17(6):8–19, 1984.

[147] Li Wern Chew, Li-Minn Ang, and Kah Phooi Seng. Survey of image compression algorithms in wireless sensor networks. In *Information Technology, 2008. ITSim 2008. International Symposium on*, volume 4, pages 1–9, 2008.

[148] Davide Brunelli, Massimo Maggiorotti, Luca Benini, and FabioLuigi Bellifemine. Analysis of Audio Streaming Capability of Zigbee Networks. In Roberto Verdone, editor, *Wireless Sensor Networks*, volume 4913 of *Lecture Notes in Computer Science*, pages 189–204. Springer Berlin Heidelberg, 2008.

[149] Sundeep Pattem, Bhaskar Krishnamachari, and Ramesh Govindan. The impact of spatial correlation on routing with compression in wireless sensor networks. *ACM Trans. Sen. Netw.*, 4(4):24:1–24:33, September 2008.

[150] Matthias Gietzelt, Stephan Schnabel, Klaus-Hendrik Wolf, Felix Büsching, Bianying Song, Stefan Rust, and Michael Marschollek. A method to align the coordinate system of accelerometers to the axes of a human body: The depitch algorithm. *Computer Methods and Programs in Biomedicine*, 106(2):97 – 103, 2012.

[151] J. Mogul, B. Krishnamurthy, F. Douglis, A. Feldmann, Y. Goland, A. van Hoff, and D. Hellerstein. Delta encoding in HTTP. RFC 3229 (Proposed Standard), January 2002.

[152] Didier Le Gall. MPEG: a video compression standard for multimedia applications. *Commun. ACM*, 34(4):46–58, April 1991.

[153] Felix Büsching, Sebastian Schildt, and Lars Wolf. DroidCluster: Towards Smartphone Cluster Computing - The Streets are Paved with Potential Computer Clusters -. In *Proceedings of PhoneCom, IEEE ICDCS 2012 International Workshop on Sensing, Networking, and Computing with Smartphones*, 2012.

[154] Hossein Falaki, Ratul Mahajan, Srikanth Kandula, Dimitrios Lymberopoulos, Ramesh Govindan, and Deborah Estrin. Diversity in smartphone usage. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys '10, pages 179–194, New York, NY, USA, 2010. ACM.

[155] Gordon E. Moore. Lithography and the future of Moore's law. volume 2438, pages 2–17, 1995.

[156] Ning Jia. AN-1023 Application Note, Fall Detection Application by Using 3-Axis Accelerometer ADXL345, 2009.

[157] Osgi Alliance. *Osgi Service Platform, Release 3*. IOS Press, Inc., 2003.

[158] Ronald E. Glaser. Bathtub and Related Failure Rate Characterizations. *Journal of the American Statistical Association*, 75(371):667–672, 1980.

[159] Lawrence Chung, BrianA. Nixon, Eric Yu, and John Mylopoulos. Introduction. In *Non-Functional Requirements in Software Engineering*, volume 5 of *International Series in Software Engineering*, pages 1–9. Springer US, 2000.

[160] J. Postel. Internet Control Message Protocol. RFC 792 (INTERNET STANDARD), September 1981. Updated by RFCs 950, 4884, 6633.

[161] D. Levi, P. Meyer, and B. Stewart. Simple Network Management Protocol (SNMP) Applications. RFC 3413 (INTERNET STANDARD), December 2002.

[162] Shashank Khanvilkar and Ashfaq Khokhar. Virtual private networks: an overview with performance evaluation. *Communications Magazine, IEEE*, 42(10):146 – 154, oct. 2004.

[163] Richard Shea. *L2Tp: Implementation and Operation*. The Addison-Wesley Networking Basics Series. Addison-Wesley, 2000.

[164] B. Patel, B. Aboba, W. Dixon, G. Zorn, and S. Booth. Securing L2TP using IPsec. RFC 3193 (Proposed Standard), November 2001.

[165] K. Hamzeh, G. Pall, W. Verthein, J. Taarud, W. Little, and G. Zorn. Point-to-Point Tunneling Protocol (PPTP). RFC 2637 (Informational), July 1999.

[166] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), August 2008. Updated by RFCs 5746, 5878, 6176.

[167] Tristan Richardson, Quentin Stafford-Fraser, Kenneth R. Wood, and Andy Hopper. Virtual network computing. *Internet Computing, IEEE*, 2(1):33 –38, jan/feb 1998.